NPS ARCHIVE
1969
WALSH, P.

# A STUDY OF DIGITAL FILTERS

Philip Joseph Walsh

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA [illegible]

United States
Naval Postgraduate School

THESIS

# A STUDY OF DIGITAL FILTERS

by

Philip Joseph Walsh

December 1969

This document has been approved for public release and sale; its distribution is unlimited.

T133760

LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF. 93940

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5101

# A Study of Digital Filters

by

Philip Joseph Walsh
Captain, United States Marine Corps
B.C.E., Marquette University, 1962

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1969

IPS ARCHIVE
969
IALSH,P.

~~[illegible]~~
c.1

## ABSTRACT

This thesis treats the subject of digital filters in two parts. First the analytical tools and synthesis techniques which are used for digital signal processors are explained and illustrated with examples. Emphasis is placed on 1) the generation and use of the digital transfer function, 2) design procedures in the frequency domain using methods of continuous-filter design as intermediate steps, and 3) real-time digital-filter implementation schemes.

In the second part of the thesis a machine-language computer program is presented for the real-time simulation of digital filters on a hybrid computer. The program is described in detail and experimental results for several filter designs and realization schemes are reported. It is believed that such a computer program, with its inherent ability to accurately simulate a special-purpose computer and to provide external control of word length and clock frequency, could be used to experimentally determine specifications for the LSI implementation of digital filters.

LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF. 93940

TABLE OF CONTENTS

## LIST OF ILLUSTRATIONS

## ACKNOWLEDGEMENT

The author wishes to thank Professor Rudolf Panholzer, Naval Postgraduate School, for his interest and assistance in the conduction of this study. Acknowledgement is also due Robert L. Limes, Supervisor of the Computer Laboratory in the Electrical Engineering Department, Naval Postgraduate School, for his extensive help in machine-language computer programming. Finally special appreciation is extended to Judith Ann Walsh, the author's wife, for her faithful diligence in typing the smooth draft of this manuscript.

# I. INTRODUCTION

The process of signal waveform filtering in linear systems may generally be defined as the application of a linear operator to an input waveform as it passes through a system. The transformation may occur in the time domain such as interpolation, extrapolation, differentiation and integration; or it may occur in the frequency domain such as low-pass filtering or smoothing, bandpass filtering, spectrum shaping and spectral decomposition. Operations such as these on continuous-time signals using analog filters are defined mathematically by linear constant-coefficient differential equations. Analysis of these operations is made possible in the s domain by the LaPlace transform, and in the f domain by the Fourier transform. Similarly linear transformations on discrete-time signals may be defined by linear constant-coefficient difference equations, and discrete-time filters may be analyzed and synthesized using the z-transform.

## A. A GENERAL DISCUSSION OF DIGITAL FILTERS

A digital filter is defined as a time-invariant linear operator on discrete-time signals. This linear operator is basically a computational process by which a sampled signal or sequence of numbers, called the input, is transformed into a second sequence of numbers, called the output. Occasionally the term "digital filter" includes the means for performing the analog-to-digital and digital-to-analog conversions.

However, since some signals are already digital in nature, "digital filter" in this paper will refer only to the computational process described above.

Digital filters are time-invariant in the sense that they are described by linear constant-coefficient difference equations. Any given set of coefficients uniquely specifies a filter, and any change in these coefficients would necessarily result in a change in the specified filter. Thus, a digital filter is time-invariant.

Implementation of the computational processes required in digital filters is accomplished using digital components such as shift registers, adders and gates. Several fundamental advantages to be gained using digital hardware are reduction in cost, higher speed, and increased reliability. Digital signal-processing techniques offer the following additional advantages over analog signal processing:

1. Very predictable, drift-free performance of arbitrarily high precision can be obtained. This permits the realization of stable filters with very high Q's.
2. There are no impedance-matching problems since digital circuits are inherently isolated from each other.
3. There is no restriction on critical filter frequencies. In particular, filters for frequencies of fractions of hertz are easy to construct.
4. A greater variety of filter functions is possible since implementation is essentially a matter of numerical coefficients vice inductors and capacitors.
5. Filter response can be changed easily by varying the proper coefficients.
6. There is an intrinsic possibility of time-sharing the major filter sections. For example, a many-section filter can be implemented by building only one section and multiplexing that section by time-switching it. The filter coefficients can be changed for each section.

Alternatively, a given filter may operate on more than one input sequence by time multiplexing.

These advantages in conjunction with the potential impact of large-scale circuit integration (LSI) on real-time digital filters assures wide application for these devices in such areas as audio systems, speech processing, seismology, oceanography, radar systems and data communications.

## B. HISTORICAL BACKGROUND

The processing of discrete signals using linear filters or weighting sequences is traced back to the 1600's by Kaiser [1]. He discusses the early use of classical numerical methods and the subsequent evolution of the z-transform calculus. The theory of the z-transform and its application to the early digital signal processors are contained in the extensive literature on sampled-data control systems published in the 1950's. See, for example, Refs. 1, 2, and 3.

More recently the general-purpose digital computer became a vehicle for the simulation of continuous filters, and the development of digital-filtering techniques per se was greatly accelerated. As computer technology advanced, particularly in the areas of memory size and computational speed, the real-time implementation of digital filters on general-purpose computers became possible.[1] Now, with the advent of

[1] Indeed, the full impact of high-speed convolution via the fast Fourier transform (FFT) on real-time digital filtering is not yet known.

large-scale circuit integration, the realization of digital filters in special-purpose hardware is becoming practical and economical.

In short, the prospect of building real-time digital filters via LSI has generated considerable interest in the general field of digital signal processing. References 4 - 8 contain much of the most recently published material on this relatively new subject.

# II. MATHEMATICAL CONSIDERATIONS

The analysis and design of digital filters requires a moderate degree of familiarization with the mathematics of discrete-time signals. In this section the theory of the sampling process and the z-transform calculus is reviewed, and their relation to the digital processing of signals is presented. Particular attention is paid to the methods of generating the digital transfer function, H(z); because, as will be shown later, this operation comprises the essence of the digital-filter design problem.

## A. THE SAMPLING PROCESS

The operation of sampling a continuous-time signal can be thought of as a form of impulse modulation. If a signal f(t) is sampled every T seconds, then the sampling rate is $\omega_s = 2\pi/T$, and the sampled signal is a train of equally spaced impulses with varying amplitude given by

$$f^*(t) = \sum_{n=o}^{\infty} f(nT)\,\delta(t-nT). \qquad (2\text{-}1)$$

In the analog-to-digital converter the amplitude of f(t) at each sample time, f(nT), is "measured," and a digital word is then formed by encoding a sequence of binary digits corresponding to that measured value. It is immediately seen that the accuracy of the "measurement" will be limited by the length of the computer word which the digital filter is designed to handle. This phenomenon is known as quantization [4, 5], and its effects are discussed in Section IV, D.

Returning to Eq. (2-1), it is evident that, in the time domain, the sampled function $f^*(t)$ equals the original function multiplied by the modulating function, $\delta(t-nT)$. Consequently, in the frequency domain, the spectrum of $f^*(t)$ will be equal to the convolution of an infinite number of equally spaced spectral lines with the original signal spectrum. Hence, the periodic spectrum shown in Fig. 2-1 is obtained. It is important to note that in a real sampler the modulating pulses are not mathematical delta functions, but pulses with a finite width. This explains the decreasing amplitude in the spectra along the frequency axis on either side of zero frequency. The shape of this attenuating function is the familiar sin x/x shape, and its derivation can be formalized using the Fourier transform.

Of particular significance is the phenomenon of frequency folding or aliasing. Again refer to Fig. 2-1. If $\omega_c$ is the highest frequency component of the original signal spectrum and $\omega_s < 2\omega_c$, distortion will appear in the sampled signal spectrum because of the overlapping of signal components. In general, therefore, in order to recover the original spectrum with a low-pass filter, that spectrum must be bandlimited to $\pm\omega_s/2$. This is known as Nyquist's Sampling Theorem [1-3] and $2\omega_c$ is called the Nyquist frequency (minimum sampling frequency). In practice there will always be some degree of frequency folding. Hence the total amount of information is never recoverable.

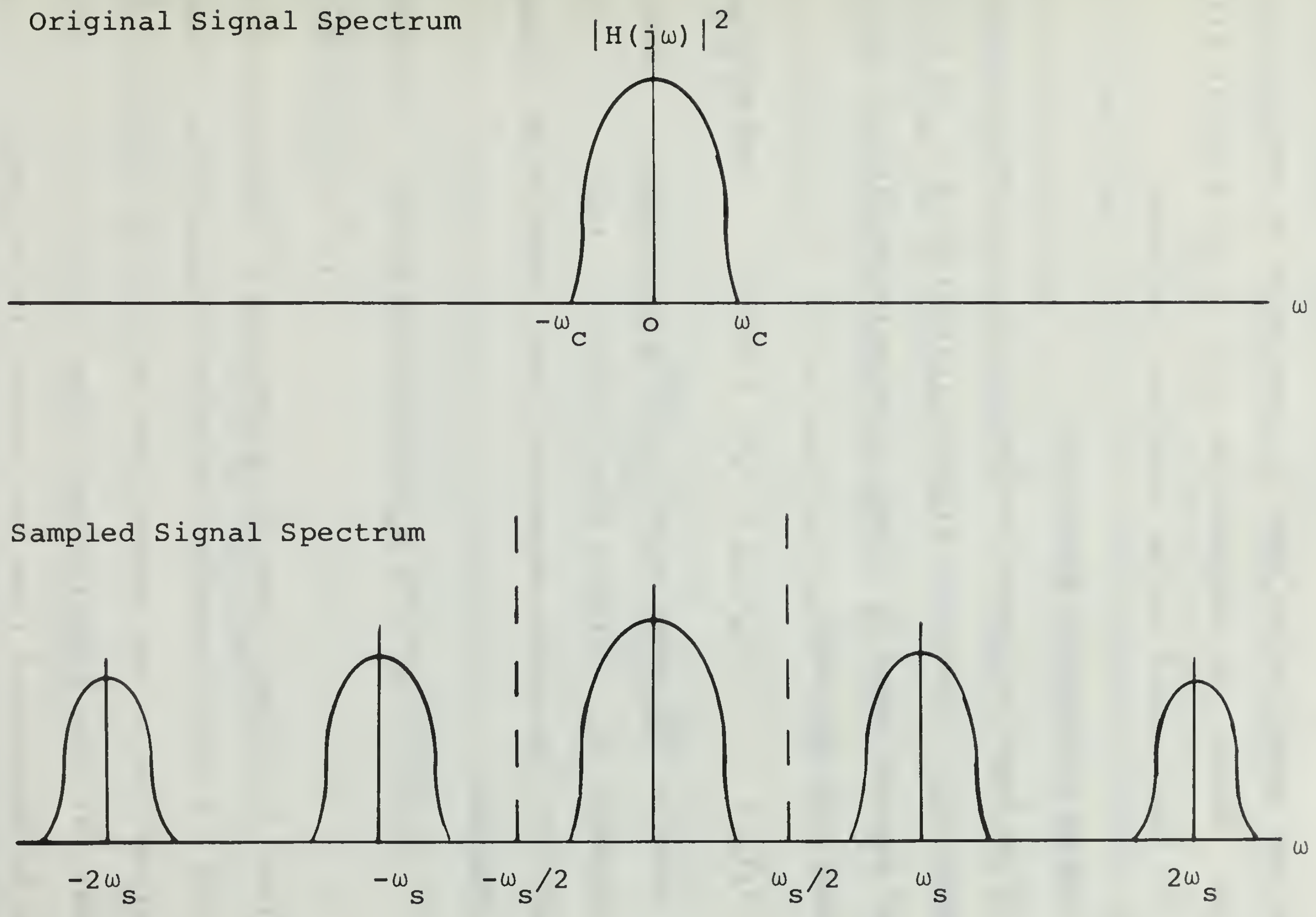


Figure 2-1. Spectrum of a Uniformly-Sampled Signal

It should be emphasized at this point that the sampling frequency must be the clock frequency for the digital filter. That is, all numerical computation that is required to generate one output value must be completed in one sampling period. Hence, the computation time of the digital filter sets an upper limit on the sampling frequency.

## B. LINEAR DIFFERENCE EQUATIONS

If an input sequence is denoted by $x_n$, and an output sequence by $y_n$, then the digital filtering of $x_n$ to produce $y_n$ may be expressed by linear constant-coefficient difference equations of the form

$$y_n + b_1 y_{n-1} + \cdots + b_M y_{n-M} = a_o x_n + a_1 x_{n-1} + \cdots + a_N x_{n-N}. \quad (2-2)$$

Each term in the above equation is a sequence of numbers. The sequence $y_{n-1}$ differs from $y_n$ only in that it is delayed one unit (T) in time. Equation (2-2), written as a function of T, becomes

$$y(nT) + b_1 y(nT-T) + \cdots + b_M y(nT-MT) = a_o x(nT) + a_1 x(nT-T) + \cdots + a_N x(nT-NT). \quad (2-3)$$

Equations (2-2) and (2-3) are equivalent. The notation of Eq. (2-2) is often preferred because of simplicity. The interpretation of the generalized difference equation is as follows: at time t=nT the new value in the output sequence $y_n$ can be determined from the current input sample, the most recent value in the input sequence $x_n$, and a linear combination of the past input and output values which are available for computation. The constant coefficients, $a_i$ and $b_i$, are determined by the particular filtering function desired. The specification of these coefficients becomes the essence of the digital-filter design problem.

An example of a first-order linear difference equation is

$$y(nT) = b_1 y(nT-T) + x(nT). \qquad (2-4)$$

(Here first-order refers to the fact that the highest order delay in the equation is one.) If $x(t)$ is the complex sinusoid $e^{j\omega t}$, then $x(nT)$ is $e^{jn\omega T}$. Accept, for the moment, that $y(nT-T)$ can be expressed as $y(nT)e^{-j\omega T}$; i.e., $y(nT-T)$ is $y(nT)$ delayed one sampling period in time. (This idea will be formalized in the next section.) Then Eq. (2-4) can be written as

$$y(nT) = \frac{e^{jn\omega T}}{1 - b_1 e^{-j\omega T}} = \frac{x(nT)e^{j\omega T}}{e^{j\omega T} - b_1}. \qquad (2-5)$$

Equation (2-5) shows that, in the steady state, the output is also a complex sinusoid, but modified by a "transfer function." The transfer function can be defined as $H(e^{j\omega T})$. Its magnitude is

$$|H(e^{j\omega T})| = \frac{1}{(1 + b_1^2 - b_1 \cos\omega T)^{1/2}} \qquad (2-6)$$

and the phase is

$$\Psi = \omega T - \tan^{-1}\left(\frac{\sin\omega T}{\cos\omega T - b_1}\right). \qquad (2-7)$$

Equation (2-6) illustrates the frequency-selective properties of the transfer function, $H(e^{j\omega T})$. Note that the magnitude response is periodic in frequency with a period equal to $\omega_s = 2\pi/T$. These results are, in fact, quite general and the concept of a discrete-signal transfer function will be discussed in detail in the next section.

## C. THE Z-TRANSFORM

It has been stated that digital filtering involves the computational process of obtaining a solution to linear constant-coefficient difference equations of the form given in Eq. (2.2). Furthermore, it was implied that the steady-state solution to Eq. (2-2) for a sinusoidal input can be obtained by applying a transfer function to the input signal. Transformations which permit the formulation of a discrete-time signal transfer function are called sampled-data or z-transforms. The term "digital transfer function" will be used to describe a discrete-time transfer function of the complex variable z.

### 1. Definition of the Z-Transform

Given a discrete signal f, represented by a sequence of numbers $f_n$; its z-transform, F(z), is defined by the power series

$$F(z) \overset{\Delta}{=} \sum_{n=o}^{\infty} f_n z^{-n} , \qquad (2-8)$$

where z is interpreted as a complex transform variable. The variable z plays a role similar to that of the LaPlace transform variable s. The operation of taking the z-transform of a sequence is denoted by

$$F(z) = Z(f_n). \qquad (2-9)$$

As a simple example, if $f_n$ were the constant signal

$$f_n = 1, n = o, 1, 2, \cdots , \qquad (2\text{-}10)$$

then the z-transform is found from Eq. (2-8) to be the geometric series

$$Z(1) = \sum_{n=o}^{\infty} z^{-n} = \frac{1}{1 - z^{-1}}, |z|>|. \qquad (2\text{-}11)$$

The theory of the z-transform is well documented [2,3]; detailed proofs and further examples will not be given here. Rather, an important theorem and several conclusions which apply to digital filters are the following:

1. The z-transform of a delayed sequence can be derived from the definition. The result is

$$Z(f_{n-k}) = z^{-k}Z(f_n). \qquad (2\text{-}12)$$

2. The z-transform of the real part of a complex sinusoid is

$$Z(a^n\cos nb) = \frac{1-a \cos b\ z^{-1}}{(1-ae^{jb}z^{-1})(1-ae^{-jb}z^{-1})}, \ |z|>a\ . \qquad (2\text{-}13)$$

3. The z-transform of the imaginary part of a complex sinusoid is

$$Z(a^n\sin nb) = \frac{a \sin b\ z^{-1}}{(1-ae^{jb}z^{-1})(1-ae^{-jb}z^{-1})}. \qquad (2\text{-}14)$$

Note that the z-transforms of sinusoids yield poles in complex-conjugate pairs at a radius depending on the damping factor.

4. The z-transform of any function of the general form

$$f_n = n^k a^n \sin(nb + c) \qquad (2\text{-}15)$$

and linear combinations of these will always be a rational function of $z^{-1}$, and such functions may be broken down into terms of the form of Eqs. (2-13) and (2-14). This class of signals corresponds to the class of continuous-time signals with rational LaPlace transforms.

The correspondence between the z-transform and the LaPlace transform will now be shown. From Eq. (2-1) a sampled signal is

$$f^*(t) = \sum_{n=o}^{\infty} f(nT)\delta(t-nT). \tag{2-16}$$

Taking the Laplace transform of both sides of Eq. (2-16) yields

$$F^*(s) = \sum_{n=o}^{\infty} f(nT)e^{-nTs} . \tag{2-17}$$

This result is identical in form with the definition of the z-transform, Eq. (2-8), provided the substitution,

$$z^{-1} = e^{-Ts} , \tag{2-18}$$

is made. The complex variable $z^{-1}$ may be interpreted as a "unit delay operator" in Eq. (2-8); or, in a different sense, as an ordering variable whose exponent represents the position of a pulse in a sequence. This latter interpretation, sometimes referred to as a "generating function" [2], will become clear after the discussion of the z-transform inverse.

2. The Digital Transfer Function

The output of a digital filter is again written as

$$y_n + b_1y_{n-1} + \cdots + b_My_{n-M} = a_ox_n + a_1x_{n-1} + \cdots + a_Nx_{n-N} . \tag{2-19}$$

Taking the z-transform of each term and applying the delayed sequence theorem gives

$$Z(y_n) + b_1 z^{-1} Z(y_n) + \cdots + b_M z^{-M} Z(y_n) = a_o Z(x_n) + a_1 z^{-1} Z(x_n) + \cdots + a_N z^{-N} Z(x_n). \tag{2-20}$$

Factoring the common terms and simplifying Eq. (2-20), the result is

$$\frac{Y(z)}{X(z)} = \frac{\sum_{i=o}^{N} a_i z^{-i}}{1 + \sum_{i=1}^{M} b_i z^{-i}} = H(z). \tag{2-21}$$

H(z) is defined as the ratio of the z-transformed output to the z-transformed input and will generally be called a digital transfer function.

Another approach to the formulation of a digital transfer function is one using the notation of a linear operator H. If an output sequence is given by

$$y_n = a_o x_n + a_1 x_{n-1} + \cdots , \tag{2-22}$$

which is simply the input sequence multiplied by a weighting function, then the notation can be simplified to

$$y_n = H(x_k), \; k = n, n-1, \cdots , \tag{2-23}$$

where H defines the filter. If $h_n$ represents the impulse response of the filter, (i.e., the response to an input sequence which is unity at n=0 and is zero at all other times), then the time response $y_n$ is obtained from the discrete convolution theorem [4],

$$y(nT) = \sum_{k=o}^{n} x(kT)h(nT-kT) \ ; \qquad (2-24)$$

or, equivalently,

$$y_n = \sum_{k=o}^{\infty} x_k h_{n-k} \ . \qquad (2-25)$$

Taking the z-transform of both sides of Eq. (2-25) and again using the delayed-sequence theorem,

$$Z(y_n) = x_o Z(h_n) + x_1 z^{-1} Z(h_n) + \cdots ,$$

or

$$Y(z) = H(z) \sum_{k=o}^{\infty} x_k z^{-k} = H(z)X(z) \qquad (2-26)$$

Again, H(z) is defined as the digital transfer function of the digital filter H.

## 3. The Z-Transform Inverse

The transform of the output sequence Y(z) is seen to be obtained from the relation Y(z)=H(z)X(z). Assuming that both H(z) and X(z) are known and are rational polynomials in z, there remains the problem of determining the sequence $Y_n$ from Y(z). Generally there are three methods for determining the inverse of a z-transform [2,3]. The first two are the use of a z-transform inversion formula and the partial fraction expansion. The inversion formula is the most general, the partial fraction expansion requiring factorization of the denominator polynomial into terms for which transform pairs are known.

A third method, and usually the easiest, is the power series or long division method. Consider the function Y(z) written in terms of $z^{-1}$ as

$$Y(z) = \frac{a_o + a_1 z^{-1} + a_2 z^{-2} + \cdots}{b_o + b_1 z^{-1} + b_2 z^{-2} \quad \cdots} . \qquad (2\text{-}27)$$

If the numerator is divided by the denominator using long division, the result is

$$Y(z) = q_o + q_1 z^{-1} + q_2 z^{-2} + \cdots . \qquad (2\text{-}28)$$

This power series gives the output directly.

$$y^*(t) = q_o(t) + q_1(t-T) + q_2(t-2T) + \cdots . \qquad (2\text{-}29)$$

Here the variable $z^{-1}$ is interpreted as the output sequence "generating function" mentioned earlier. Such a routine is easily performed on a computer or calculating machine.

## 4. Sinusoidal-Steady-State Solution of Linear Difference Equations

In continuous-time linear systems the labor of taking inverse LaPlace or Fourier transforms can be avoided for the special case of sinusoidal input functions. The output of such a system in the steady state will always be a sinusoid of the same frequency, but whose magnitude and phase are modified by the transfer function of the system. The solution for the magnitude of the output over a specified frequency range is called the frequency response of the system. This function is found by evaluating the system transfer function H(s) or F(jω) along the real frequency axis. It will now be shown that the response of a discrete-time linear system can be found by evaluating the digital transfer function H(z) around the unit circle in the z-plane. Hence, for the very common case of an input sequence being a uniformly sampled sine wave, the output sequence may be determined without calculating a z-transform inverse.

In Section II, B Eq. (2-5) showed that the output of a discrete first-order system to a sinusoidal input was the same sinusoid modified by a frequency-sensitive transfer function $H(e^{j\omega T})$. Applying the substitution $z=e^{sT}$, with $s=j\omega$, permits the transfer function to be written as a function of z.

$$H(e^{j\omega T}) = \frac{e^{j\omega T}}{e^{j\omega T} - b_1} = \frac{1}{1 - b_1 z^{-1}} = H(z). \qquad (2\text{-}30)$$

If the original difference equation, Eq. (2-4), had been solved using the z-transform, the resultant digital transfer function would have been identical to the one in Eq. (2-30) above. These results are not just coincidental, but are required by the definition of the z-transform. Therefore, it can be stated generally that the frequency response of a digital filter[2] can be found by evaluating the digital transfer function H(z) of that filter at $z=e^{j\omega T}$.

This discussion is illustrated graphically in Fig. 2-2. The mapping of the s-plane into the z-plane is accomplished by substituting $s=\sigma+j\omega$ into Eq. (2-18). Then

$$z = e^{(\sigma+j\omega)T} = e^{\sigma T} \cdot e^{j\omega T}, \qquad (2\text{-}31)$$

[2]In this context the term digital filter is intended to refer only to frequency-selective filters in the analog sense, vice the general time-domain filters mentioned in Section I, A.

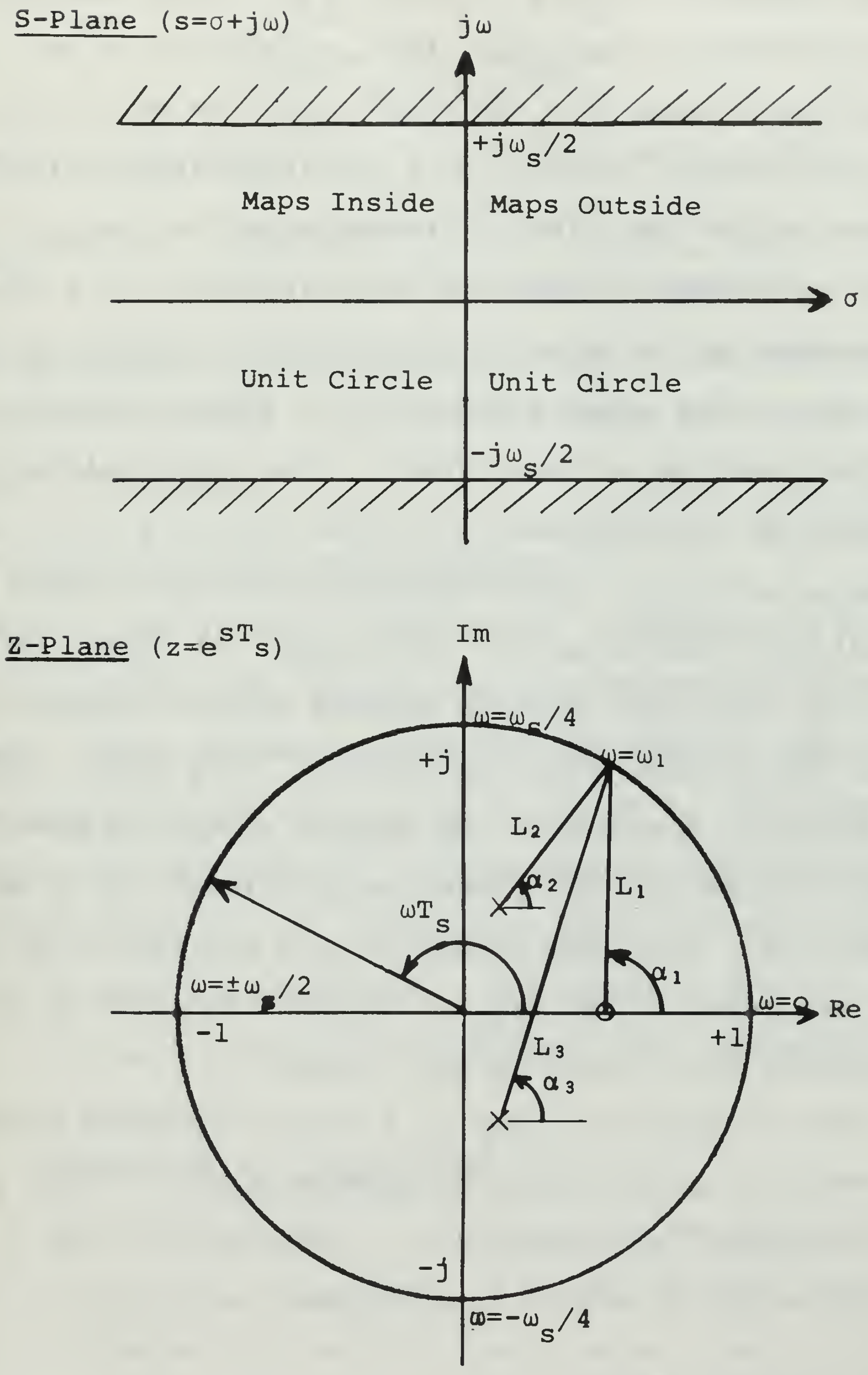


Figure 2-2. S-Plane to Z-Plane Transformation

limit

where $T=2\pi/\omega_s$ For $|\omega|<\omega_s/2$ the right portion of the s-plane (positive $\sigma$) maps into the area outside the unit circle in the z-plane. Similarly, the left portion of the s-plane (negative $\sigma$) maps into the area inside the unit circle. Just as the frequency response of a continuous-time filter is determined by the value of its transfer function on the $j\omega$ axis, the frequency response of a discrete-time filter is determined by the value of its transfer function on the unit circle. If the poles and zeros of a digital transfer function are plotted in the z-plane,[3] then the magnitude of the response at the frequency $\omega_1$ in Fig. 2-2 is $L_1/L_2L_3$ and the phase is $\alpha_1-\alpha_2-\alpha_3$. These results are obtained using conventional pole-zero graphical techniques. It should be noted that the magnitude response repeats after $\omega$ reaches $\omega_s$, $2\omega_s$, etc., and is symmetrical around $n\omega_s$. This result agrees with the periodic spectrum of the sampled signal discussed previously. In general the phase characteristic may or may not repeat [9]. Also, the stability of a digital filter requires that the poles of the digital transfer function be located within the unit circle in the z-plane.

The frequency response of a digital transfer function can be evaluated analytically by substituting $e^{-j\omega T}$ for $z^{-1}$ in the functional expression $H(z)$. Assuming $H(z)$ can be factored into products of second-order expressions, then

[3] Poles and zeros of a digital transfer function are found, as expected, by the factorization of the rational polynomials in z.

$$H(e^{-j\omega T}) = \frac{A_o + A_1 e^{-j\omega T} + A_2 e^{-j2\omega T}}{1 + B_1 e^{-j\omega T} + B_2 e^{-j2\omega T}}$$

$$= \frac{A_o e^{j\omega T} + A_1 + A_2 e^{-j\omega T}}{e^{j\omega T} + B_1 + B_2 e^{-j\omega T}} .$$

Expanding the exponential terms and regrouping yields

$$H(e^{-j\omega T}) = \frac{[(A_2+A_o)\cos\omega T+A_1]+j[(A_o-A_2)\sin\omega T]}{[(B_2+1)\cos\omega T+B_1)]+j[(1-B_2)\sin\omega T]} . \qquad (2\text{-}32)$$

Computer Program 1 may be used to evaluate the magnitude of the right side of Eq. (2-32). The program is written in the FORTRAN IV language.

## D. S-PLANE TO Z-PLANE TRANSFORMATIONS ✓

Many of the digital-filter design techniques to be discussed in Section III involve the digitalization of analog filters. These indirect design methods require transformations from functions of s to functions of z. This section discusses the mechanics of various mapping transformations from the s-plane to the z-plane and compares the relative merits of each transformation. The formulas given for computation of coefficients are not derived, but reproduced from the literature [6].

### 1. The Standard Z-Transform

This transformation is simply the extension of the definition of the z-transform. A proper rational function of s is expanded into partial fractions and the z-transform of each term is calculated. If RR represents the real part of a residue in the partial fraction expansion, and RI represents the imaginary part, then real terms are transformed according to

$$\frac{RR}{s - u} \rightarrow \frac{A_o}{1 + B_1 z^{-1}} , \tag{2-33}$$

where

$$A_o = T(RR)$$
$$B_1 = - \exp(uT) .$$

Complex conjugate terms may be combined and transformed according to the transform pair

$$\frac{RR + jRI}{s - u - jv} + \frac{RR - jRI}{s - u + jv} \rightarrow \frac{A_o + A_1 z^{-1}}{1 + B_1 z^{-1} + B_2 z^{-2}} , \tag{2-34}$$

where

$$A_o = 2T(RR)$$
$$A_1 = 2T \exp(uT)[RR \cos(vT)+RI \sin(vT)]$$
$$B_1 = 2 \exp(uT)\cos(vT)$$
$$B_2 = \exp(2uT) .$$

Since the transformation involves the relation, $z^{-1} = e^{-sT}$, the mapping of the s-plane into the z-plane takes place exactly as was previously described for Fig. 2-2. Notice that the transformation maps each strip in the s-plane, bounded by $\omega = (n+1/2)\omega_s$ and $\omega = (n-1/2)\omega_s$, where n is an integer, into the entire z-plane. Thus, the mapping of each succeeding strip is superimposed on the baseband mapping function. If the transfer function H(s) has any significant response above $\omega = \omega_s/2$ (i.e., if the product of the zero vectors divided by the product of the pole vectors is not negligible for $\omega>\omega_s/2$, frequency folding will occur and the transformation may be unsatisfactory. Consequently the standard z-transform is normally used only for bandlimited functions. The problem is related to the one discussed for the sampled signal. In the present case, however, the transfer function itself must be bandlimited in addition to the input signal. In cases where H(z) is not sufficiently bandlimited a low-pass "guard" filter may be inserted in cascade to permit a successful transformation.

2. The Bilinear Z-Transform

To circumvent the problem of frequency folding a transformation is sought which will map the entire s-plane into the z-plane. As a first step, the entire s-plane is mapped into an s'-plane bounded by the lines $s' = \pm j\omega_s/2$. The mapping relation is

$$s = \frac{2}{T} \tanh \left(\frac{s'T}{2}\right). \tag{2-35}$$

Note that this transformation also maps the entire s-plane into each succeeding strip bounded by $\omega' = (n+\frac{1}{2})\omega'_s$ and $\omega' = (n-\frac{1}{2})\omega'_s$ for integer values of n. This condition is necessary since the subsequent function of z must be periodic in frequency. Then, substituting $z = e^{-s'T}$ into Eq. (2-35), the bilinear z-transform becomes

$$s = \frac{2}{T} \tanh \left(\frac{s'T}{2}\right) = \frac{2\ (1-e^{-s'T})}{T\ (1+e^{-s'T})} = \frac{2\ (1-z^{-1})}{T\ (1+z^{-1})} . \tag{2-36}$$

This algebraic transformation does map the entire left half of the s-plane into the interior of the unit circle in the z-plane. Folding errors are eliminated since no folding occurs. The price paid for this feature is the nonlinear warping of the frequency scale. If $s' = j\omega'$ and $s = j\omega$ in Eq. (2-35), then

$$\frac{\omega T}{2} = \tan(\frac{\omega' T}{2}) \quad . \tag{2-37}$$

Since $\omega'$ is the frequency which is really mapped into the z-plane, the warping of the frequency scale according to the tangent function results. However, satisfactory results can usually be obtained by prewarping the design frequency before performing the transformation.

3. The Matched Z-Transform

This transform generates a digital transfer function with poles and zeros matched to those of the continuous function. The mapping transformation is

$$s \rightarrow e^{sT} = z \quad . \tag{2-38}$$

Then real poles or zeros transform according to

$$s - u_1 \rightarrow 1 + A_1 z^{-1} \quad , \tag{2-39}$$

where

$A_1 = - \exp(uT)$

Complex poles or zeros are combined into second-order factors and transformed according to

$$(s-u)^2 + v^2 \rightarrow 1 + A_1 z^{-1} + A_2 z^{-2} \quad , \tag{2-40}$$

where

$A_1 = -2\exp(uT)\cos(vT)$
$A_2 = \exp(2uT)$

The matched z-transform preserves the shape of the frequence response characteristics. Golden [6] points out that it may sometimes be necessary to multiply the resultant

H(z) by terms of the form $(1 + z^{-1})^n$ to achieve satisfactory results. This operation is equivalent to inserting n additional zeros at the half-sampling frequency into the digital transfer function.

4. Other Transformations

Kaiser [1] discusses various other s-plane to z-plane transformations which may be used to achieve specific results. One particular transformation of interest is

$$s \rightarrow \frac{z^2 - 2z \cos\omega_o T + 1}{z^2 - 1} . \qquad (2\text{-}41)$$

This transformation, which is a variation of the bilinear z-transform, maps the entire imaginary axis of the s-plane onto both the top and bottom arcs of the unit circle in the z-plane. Setting s = 0 and solving the numerator of (2-41) for z shows that the origin of the s-plane is mapped into the points $e \pm j\omega_o T$. Thus, applying this transformation to a low-pass H(s) yields a digital bandpass H(z) with center frequency of $\omega_o$. If we denote the analog cutoff frequency by $\omega_A$ and the digital cutoff frequency by $\omega_D$, then (2-41) can be written in equation form as

$$j\omega_A = \frac{e^{j2\omega_o T} - 2e^{j\omega_D T} \cos\omega_o T + 1}{e^{j2\omega T} - 1} ,$$

or

$$\omega_A = \frac{\cos\omega_o T - \cos\omega_D T}{\sin\omega_D T} . \qquad (2\text{-}42)$$

Again frequency warping is evident; but, as with the bilinear z-transform, prewarping may be used to produce useful results.

5. A Summary of S-Plane to Z-Plane Transformations

Each of the transformations discussed in this section has advantages and disadvantages. In the design of digital filters the choice of a transformation must be dictated by these factors.

First the standard z-transform, which is an exponential transform, requires the partial fraction expansion of the transfer function H(s) and the attendant factorization of the denominator polynomial. The transform does preserve the shape of the impulse time response; but, because of frequency folding, useful results can be obtained only for bandlimited functions of the low-pass and bandpass variety.

The bilinear z-transform, being an algebraic transform, is easy to implement and requires no reduction of the transfer function in s. However, because of the frequency warping, this transformation should be used only for functions with piecewise-constant magnitude characteristics. The bilinear form does preserve flat gain characteristics.

The matched z-transform, like the standard z-transform, is exponential in form, but requires only the factored form of H(s), not the partial fraction expansion. It preserves the shape of the frequency response characteristics, and gives good results for high-pass and bandstop filters.

# III. DIGITAL-FILTER DESIGN TECHNIQUES

It has been shown that digital filters may be described by digital transfer functions of the form

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=o}^{N} a_i z^{-i}}{1 + \sum_{i=1}^{M} b_i z^{-i}} . \qquad (3\text{-}1)$$

Equation (3-1) is perfectly general and applies to all digital-filter functions. The nature of the filter function is defined completely by the $a_i$ and $b_i$, and the specifications of these coefficients becomes almost the entire filter-design problem.

Digital filters are generally classified as being "recursive" or "non-recursive." Referring to Eq. (3-1) if at least one of the $b_i$ are non-zero, the filter is said to be of the recursive type; i.e., the computation of the present value of the output requires not only the present and past values of the input, but also the past values of the output (feedback). If all the $b_i$ are zero, then the filter is said to be of the non-recursive or transversal type; i.e., the output is simply a linear combination of weighted input values. This filter is also referred to as a "tapped delay line" filter.

The design methods for the two classes of filters differ markedly and each class has its distinct properties. The non-recursive filter generally requires a large number of terms to obtain relatively sharp transitions in frequency response. However, the non-recursive filters have excellent phase characteristics, being quite linear in some cases. In contrast, the recursive filter can provide sharp cutoff characteristics with relatively few terms, but the phase characteristics are poor.

In Section IV it will be shown that real-time implementation of digital filters is presently much easier and more practical when the recursive filter structure is used. For this reason the discussion of non-recursive filters in this thesis is brief. All digital-filter design examples are recursive type filters to permit subsequent real-time implementation.

## A. NON-RECURSIVE DIGITAL FILTERS

The design methods for non-recursive filters fall into three categories. The first category is the classical numerical approach. In this method classical interpolation and differentiating formulas for equally spaced data can be digitalized directly by making correspondence between the unit advance operator "E" in numerical analysis and the z-transform operator. These filters are good only at low frequencies and they correspond to using the first few terms of a power-series expansion. This method is restricted primarily to filters whose function is integration, interpolation, differentiation, etc.

The second design category involves the use of the Fourier series. If the magnitude of the desired frequency response is expanded in a Fourier series over $|\omega|<\omega_s/2$ and the series is written in terms of cosines (or sines) the result is

$$H(\omega) = \sum_{n=o}^{\infty} a_n \cos(n\omega T) . \qquad (3\text{-}2)$$

But, since $z^{-1} = e^{-j\omega T}$, Eq. (3-2) can be written as

$$H(z) = a_o + \frac{1}{2} \sum_{n=1}^{\infty} a_n (z^n + z^{-n}) , \qquad (3\text{-}3)$$

which is the required filter. The filter has an infinite number of terms and is realizable only if an approximation is made by truncating the series. The approximation is good for a rapidly converging series, but will be poor if there are discontinuities or steep transitions in the desired frequency response.

The third category of design methods for non-recursive filters involve attempts to achieve a better convergent series, or to systematically weight error functions. Methods exist which utilize the Chebyshev series, least-squares approximation and "window" functions. Kaiser [1] treats this subject extensively and he presents several examples illustrating the design methods.

## [illegible]. RECURSIVE DIGITAL FILTERS

Recursive digital-filter designs may be accomplished either in the time domain or in the frequency domain. Historically designers of sampled-data filters were concerned with step-response, peak over-shoot, etc., and most transfer functions were analyzed and synthesized to satisfy time-domain criteria such as these. More recently, since the scope of digital signal processors has widened, more emphasis has been placed on steady-state response in the frequency domain. ~~Gold and Rader [4] have formalized various proce-dures for digital-filter design in the frequency domain. Golden [6] has also contributed to the development of these methods; furthermore, he has written a computer program which essentially performs all the design computations to be found in this section. The program was not available to the author of this thesis.~~

There are two general approaches to the design of recursive filters. They are the direct design method and the indirect design method. This latter method, which appears to be more popular, uses many of the standard continuous-time filters as intermediate designs.

1. <u>Direct Design Method</u>

This method employs computational techniques to determine the coefficients, $a_i$ and $b_i$, directly from the filter specifications. Some methods involve the use of polynomial-approximation theory. Another technique is to locate poles and zeros directly in the z-plane [illegible], and then possibly to perform an iteration or trial-and-error procedure. These design methods might be employed where the specified response is not of a standard form (low-pass, bandpass, etc.). ~~They will not be discussed further in this thesis~~.

2. <u>Digital-Filter Design via Analog-Filter Design</u>

The indirect method of digital-filter design is accomplished in two steps. First an analog filter H(s) is designed to satisfy the filter specifications. Then a transformation of the analog function is performed to determine a digital transfer function H(z). In this manner use can be made of the many design and synthesis techniques for continuous filters which have been developed over the years. ~~Reference 10 is an example of the extensive treatment of this subject in the literature.~~

a. Topics in Continuous-Filter Design

~~Standard forms of continuous filters have been developed and are well documented in the literature [10]~~. The filters are referred to by the names Butterworth, Chebyshev, Elliptic, Bessel, Lerner and others. Each type of filter has certain characteristics which determine the most appropriate use for that filter. Of these many filter types the Butterworth and the Chebyshev designs are now briefly outlined to provide a basis for the digital-filter design examples which follow.

(1) <u>The Butterworth Filter</u>. The Butterworth filter can be specified by the relationship

$$|H(j\omega)|^2 = \frac{1}{1 + (\omega/\omega_c)^{2n}} , \qquad (1\text{-}1)$$

where $\omega_c$ is the cutoff frequency. $|H(j\omega)|^2$ is the squared magnitude of the filter transfer function. It can be shown that the transfer function H(s) is a rational function of s with a constant numerator and a denominator polynomial of degree n. For example, a fourth-order Butterworth filter with $\omega_c$ normalized to 1 rad/sec has a transfer function

55.3

5
3
2

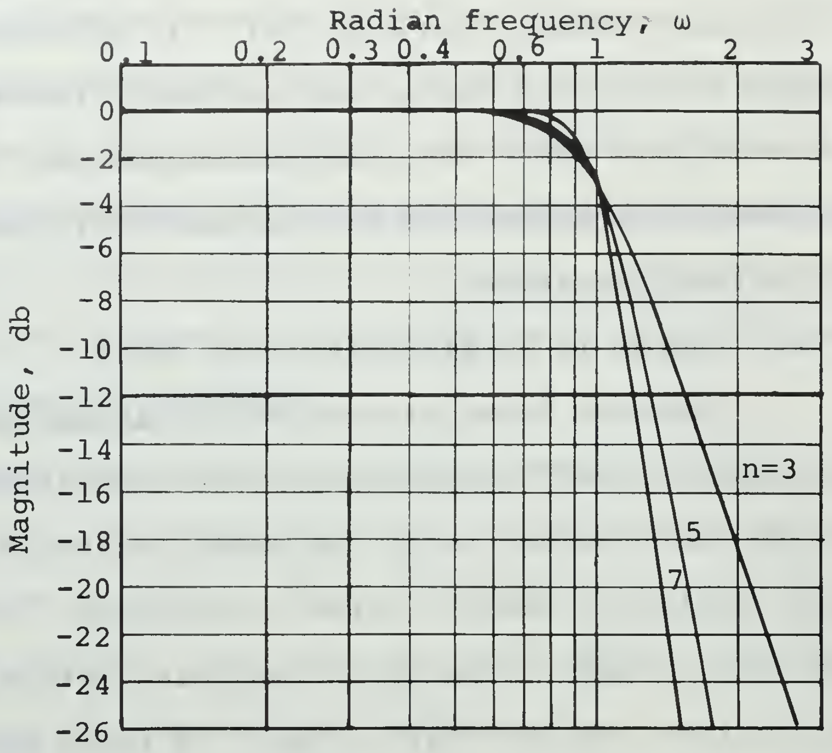


Figure 3-1. Response of a Butterworth Low-Pass Filter.

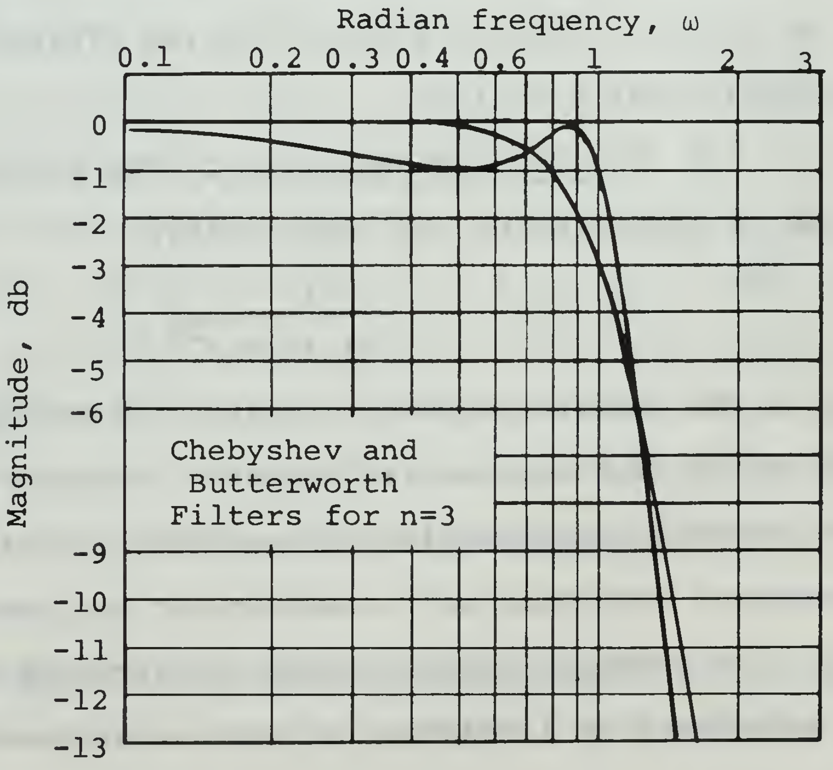


Figure 3-2. Response of a Chebyshev Low-Pass Filter.

$$H(s) = \frac{1}{s^4 + 2.613s^3 + 3.414s^2 + 2.613s + 1.} \qquad (1-2)$$

The coefficients for other-order filters are readily found in tables [10]. Plots of the Butterworth frequency response for three values of n are shown in Fig. 1-1. Generally the response is characterized by a flat passband and a stopband attenuation of 20 ndb/decade.

(2) <u>The Chebyshev Filter</u>. The Chebyshev filter is specified by

$$|H(j\omega)|^2 = \frac{1}{1 + \varepsilon^2 V_n^2(\omega/\omega_c)} , \qquad (1-3)\ \text{~~(3-6)~~}$$

where $V_n$ is a Chebyshev polynomial of order n. The Chebyshev filter is characterized by an equal ripple in the passband and a monotonic decay in the stopband. The ripple amplitude, $\delta$, is given by

$$\delta = 1 - \frac{1}{\sqrt{1 + \varepsilon^2}} . \qquad (1-4)\ \text{~~(3-7)~~}$$

The transfer function of a Chebyshev filter is also a rational function of s. The coefficients of the transfer function are determined by the specified value of $\varepsilon$ and are calculated by a somewhat involved but well established procedure [10]. A plot of the response of a typical third-order Chebyshev filter is shown in Fig. 1-2. Generally the Chebyshev filter will have a steeper slope in the stopband than a Butterworth filter of the same order.

(3) <u>Frequency Scaling</u>. Since most continuous filters are designed using normalized frequencies

(i.e., $\omega_c$ = 1 rad/sec), frequency scaling is required to shift the response of the filter up to the frequency band of interest. If we denote the desired cutoff frequency by $\omega_c$, then to shift the frequency response of the normalized transfer function to $\omega_c$ make the substitution

$$s_n = \frac{s}{\omega_c} . \qquad \text{1-5} \quad \text{~~(3-8)~~}$$

(4) Frequency Transformations. High-pass, bandpass, and bandstop filters may all be obtained from a low-pass design by effecting the following frequency transformations:

Low-pass-to-high-pass (1.6a)

$$s \rightarrow \frac{\omega_c}{s} \qquad \text{~~(3-9a)~~}$$

Low-pass-to-bandpass

$$s \rightarrow \frac{\omega_o}{BW}\left(\frac{s}{\omega_o} + \frac{\omega_o}{s}\right) \qquad (1\text{-}6b) \quad \text{~~(3-9b)~~}$$

Low-pass-to-bandstop

$$s \rightarrow \frac{BW}{\omega_o\left(\frac{s}{\omega_o} + \frac{\omega_o}{s}\right)} \qquad (1\text{-}6c) \quad \text{~~(3-9c)~~}$$

In the expressions (1-6) ~~(3-9)~~ BW = $\omega_{c2} - \omega_{c1}$ and $\omega_o = \sqrt{\omega_{c2} \cdot \omega_{c1}}$ (the geometric mean).

These transformations are all performed on the normalized low-pass H(s) since they contain their own scaling. Note that when using the bandpass and bandstop transformations $\omega_o$ is not necessarily in the center of the band. Consequently final designs may be slightly asymmetric relative to the intended center frequency.

b. The Design Procedure

A desired frequency response is normally specified by the nature of the characteristic in the passband and the amount of attenuation in the stopband. For example, it may be required to design a bandpass filter which is flat in the passband centered at 1000 Hz, down 3 db at 800 Hz and 1200 Hz (cutoff frequencies), and down at least 20 db at 500 Hz and 1500 Hz. The flat characteristic would normally call for a Butterworth design, the 20-db attenuation specification would determine the order of the filter and the 3-db frequencies would specify the required frequency transformation. In the case where a final digital-filter design is required, the sampling frequency would be an additional and extremely important specification.

The next decision to be made is the type of z-transformation to be used. Generally the standard z-transform would be used only for bandlimited functions to prevent frequency folding as discussed earlier. Bandlimited functions are typically functions of s whose denominator is of high degree and whose cutoff frequency is a small fraction of the half-sampling frequency. Usually the success of a standard z-transform design cannot be assured without actually testing the final filter.

The bilinear z-transform works well when the frequency characteristic is relatively piecewise-constant. Hence, most Butterworth designs and small-ripple Chebyshev designs could be transformed satisfactorily. The algebraic form of the bilinear z-transform makes it easy to implement; but prewarping of the critical frequencies in the s-domain is required to achieve the desired response in the z-domain.

Finally the matched z-transform may be used to achieve a frequency characteristic that is high-pass, band-stop, or other than piecewise-constant.

The digital-filter design procedures discussed above will be illustrated by examples in the next section.

## [illegible]. EXAMPLES OF RECURSIVE DIGITAL FILTERS

The following examples were chosen to illustrate the digital-filter design methods which have been discussed. The order of the filters was kept low to facilitate the handling of the numbers. All calculations were performed on an electronic calculator, and the results were rounded to the sixth decimal place. ~~For more complex filters the use of a computer program such as the one written by Golden [6] would almost certainly be required.~~

EXAMPLE 1. Design a low-pass digital filter to operate at a sampling frequency of 2000 Hz. The cut off (3-db) frequency is to be 200 Hz and the response is to be essentially flat in the passband.

1. A second-order Butterworth filter is chosen with a subsequent transformation using the bilinear z-transform. Prewarping of the analog cutoff frequency is required. From

~~Eq. (2-37)~~ 4

$$\omega_A = \tan\left(\frac{\omega_D T}{2}\right) . \qquad (1-7)^{*}$$

$$\omega_A = \tan\left(\frac{2\pi \cdot 200 \cdot .5\times10^{-3}}{2}\right) = \tan(0.314159) = 0.324920. \qquad ~~(3-10)~~$$

Note that the analog cutoff frequency $\omega_A$ has been prewarped to (0.324920)(2/T) = 207 Hz.

2. A normalized second-order Butterworth filter is

$$H(s) = \frac{1}{s^2 + 1.414213s + 1} . \qquad ~~(3-11)~~ \; (1-8)$$

Substituting s = s/0.324920 ~~in Eq. (3-11)~~ gives

$$H_1(s) = \frac{0.105573}{s^2 + 0.459506s + 0.105573} . \qquad ~~(3-12)~~ \; (1-9)$$

3. The bilinear z-transform is taken by substituting s = (z-1)/(z+1) in Eq. ~~(3-12)~~. The result is

$$H_1(z) = \frac{0.105573(z^2+2z+1)}{z^2-2z+1+0.459506(z-1)(z+1) + 0.105573\,(z^2+2z+1)} ,$$

which reduces to

$$H_1(z) = \frac{0.067455 + 0.134910\,z^{-1} + 0.067455\,z^{-2}}{1 - 1.142980\,z^{-1} + 0.412801\,z^{-2}} . \qquad (1-10)$$

The frequency response of this digital filter was evaluated using Computer Program 1, and the results are plotted ~~in~~ at the end ~~Fig. 3-3~~ of the program. The 3-db frequency is exactly 200 Hz. The poles

---

* The factor 2/T is dropped here and in the subsequent substitution for s to avoid handling large numbers. Should the factor be retained the result can always be reduced to the same final answer.

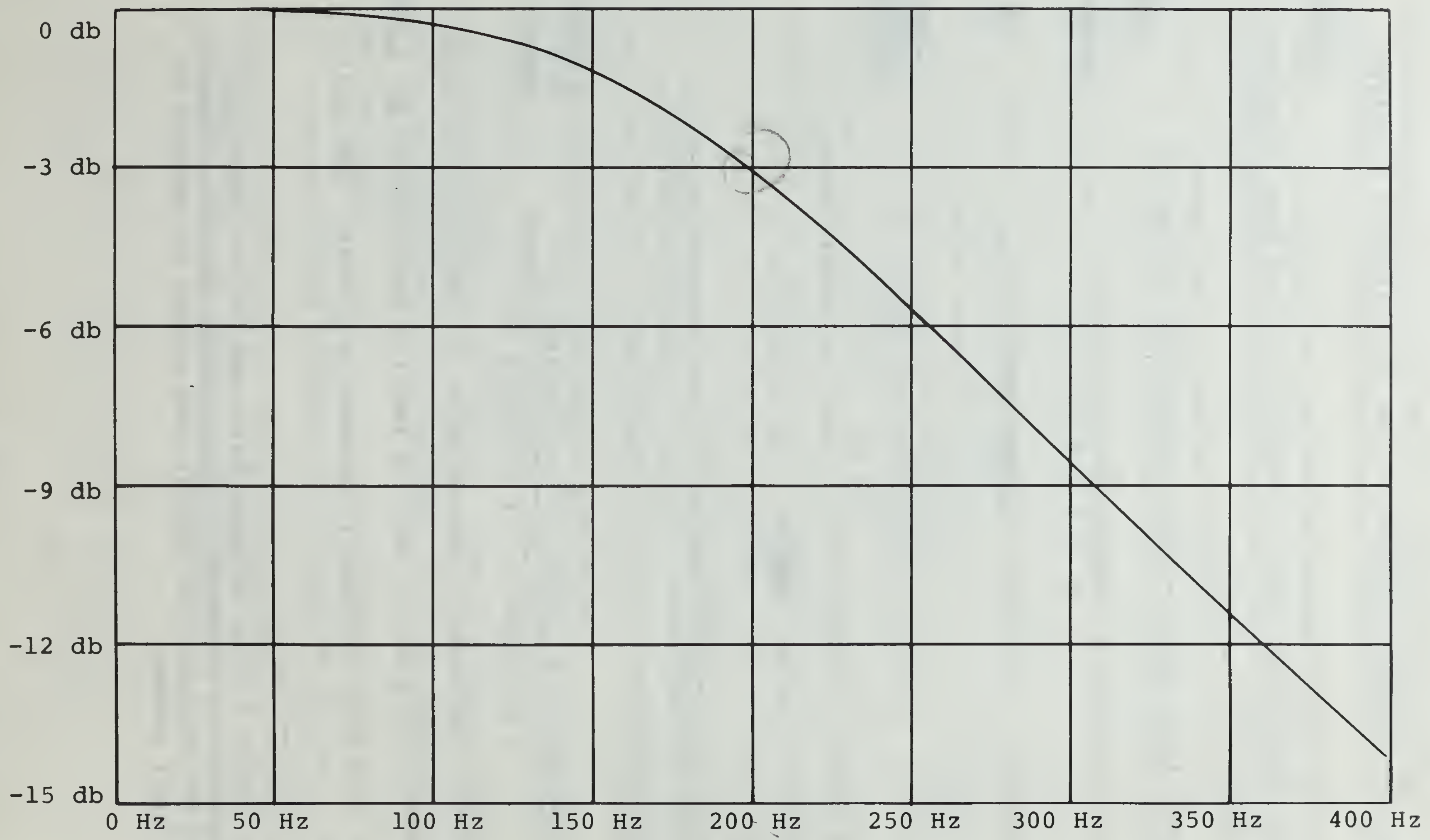


Figure 3-3. Response of a Low-Pass Digital Filter

of the transfer function, [redacted] ([redacted]) are:

$z = 0.571490 \pm j0.293598$

These poles are well inside the unit circle indicating a stable filter.

EXAMPLE 2. Design a bandpass digital filter to operate at a sampling frequency of 2000 Hz. The cutoff frequencies, $\omega_{D1}$ and $\omega_{D2}$, are 200 Hz and 500 Hz respectively, and the response is to be flat in the passband.

1. A second-order Butterworth is again chosen to be used with the bandpass bilinear z-transform given in Eq. (2-41). This transformation will yield a fourth-order function in z for a second-order function in s.

2. Equation (2-42) is written again for convenience.

$$\omega_A = \frac{\cos\omega_o T - \cos\omega_D T}{\sin\omega_D T} \tag{3-14}$$

It is desirable to have the analog frequencies associated with each of the critical digital frequencies be negatives of each other. Then the digital transfer function which results by transforming $H(s/|\omega_A|)$ will satisfy the response specifications at both critical frequencies, $\omega_{D1}$ and $\omega_{D2}$. To accomplish this solve for $\cos\omega_o T$ in

$$\frac{\cos\omega_o T - \cos\omega_{D1} T}{\sin\omega_{D1} T} = \frac{-\cos\omega_o T + \cos\omega_{D2} T}{\sin\omega_{D2} T} . \tag{3-15}$$

After considerable manipulation of Eq. (3-15) using trigonometric identities, the result is

$$\cos\omega_o T = \frac{\cos\frac{1}{2}(\omega_{D1}T + \omega_{D2}T)}{\cos\frac{1}{2}(\omega_{D1}T - \omega_{D2}T)}. \qquad (3\text{-}16)$$

Equation (3-16) need be derived only once. It then becomes a formula for use in the design procedure for the bandpass bilinear z-transform.

3. Returning to the example

$$\cos\omega_o T = \frac{\cos\frac{1}{2}(700 \cdot 2\pi \cdot T)}{\cos\frac{1}{2}(-300 \cdot 2\pi \cdot T)} = 0.509523. \quad ✓ \qquad (3\text{-}17)$$

4. Then, using Eq. (3-14),

$$\omega_{D1}T = (200)(2\pi)(0.5 \times 10^{-3}) = 0.62832,$$

and

$$\omega_A = \frac{0.509523 - \cos 0.62832}{\sin 0.62832} = 0.509523. \qquad (3\text{-}18)$$

An identical result is obtained for $\omega_{D2}T = 1.57080$ verifying the validity of Eq. (3-16).

5. Substituting s = s/0.509523 into the normalized second-order Butterworth transfer function gives

$$H(s) = \frac{0.259614}{s^2 + 0.720574s + 0.259614} \quad ✓ \qquad (3\text{-}19)$$

6. The digital transfer function is now obtained by substituting $s = (z^2 - 1.019046z + 1)/(z^2 - 1)$ into Eq. (3-19). The result after considerable algebra is

$$H(z) = \frac{0.131106-0.262212z^{-2}+0.131106z^{-4}}{1-1.400064z^{-1}+1.272216z^{-2}-0.658419z^{-3}+0.272217z^{-4}}; \qquad (3\text{-}20a)$$

or

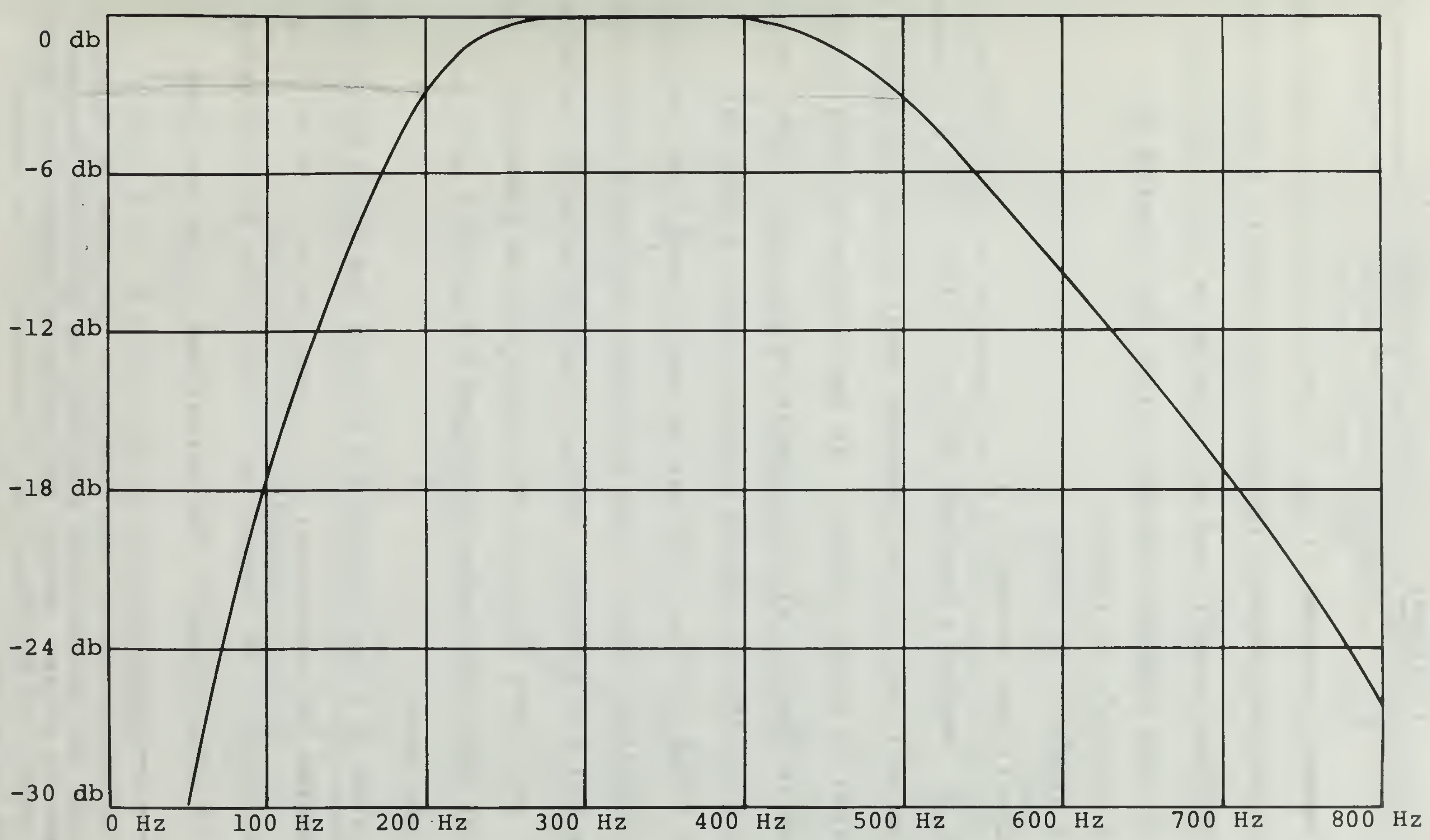


Figure 3-4. Response of a Bandpass Digital Filter.

$$H(z) = \frac{(\overset{A_0}{0.362085}-\overset{A_{11}}{0.724170}z^{-1}+\overset{A_{22}}{0.362085}z^{-2})^2}{(1-\underset{B_{11}}{1.222023}z^{-1}+\underset{B_{22}}{0.603825}z^{-2})(1-0.178041z^{-1}+0.450821z^{-2})} \quad (3\text{-}20b)$$

The frequency response of the filter is evaluated using Eq. (3-20b) and the computer program. The results satisfy the specifications exactly and are plotted in Fig. 3-4. The poles of this bandpass digital filter are located at

$z = 0.611011 \pm j0.480094$

$z = 0.089020 \pm j0.665505$

Again the filter is stable.

<u>EXAMPLE 3</u>. Design a bandpass digital filter with a frequency characteristic within the passband equal to the reciprocal of a gaussian or normal curve. (An application of this filter is given in Section VI, C.) Refer to Fig. 3-5. The center frequency is 450 Hz and the sampling rate is 2000 Hz.

1. The shape of Fig. 3.5b may be approximated by a second-order Chebyshev filter. (An nth-order Chebyshev filter has n peaks and n-1 troughs. Hence, a second-order filter with a large $\varepsilon$ will have a dip in the center of the passband.) The design of this particular Chebyshev filter will only be summarized here.

A value of $\varepsilon$ was calculated to give the dip a shape which approximates the reciprocal of a normal curve. To do this a relation between the statistics of the curve and frequency had to be established.[5] The normalized filter was then

[5] In Section VI, C, the origin of the gaussian curve is seen to be the spectrum of a CW carrier which is frequency modulated by noise. Hence, the relation between the spread of the curve and frequency is determined by the actual hardware being used.

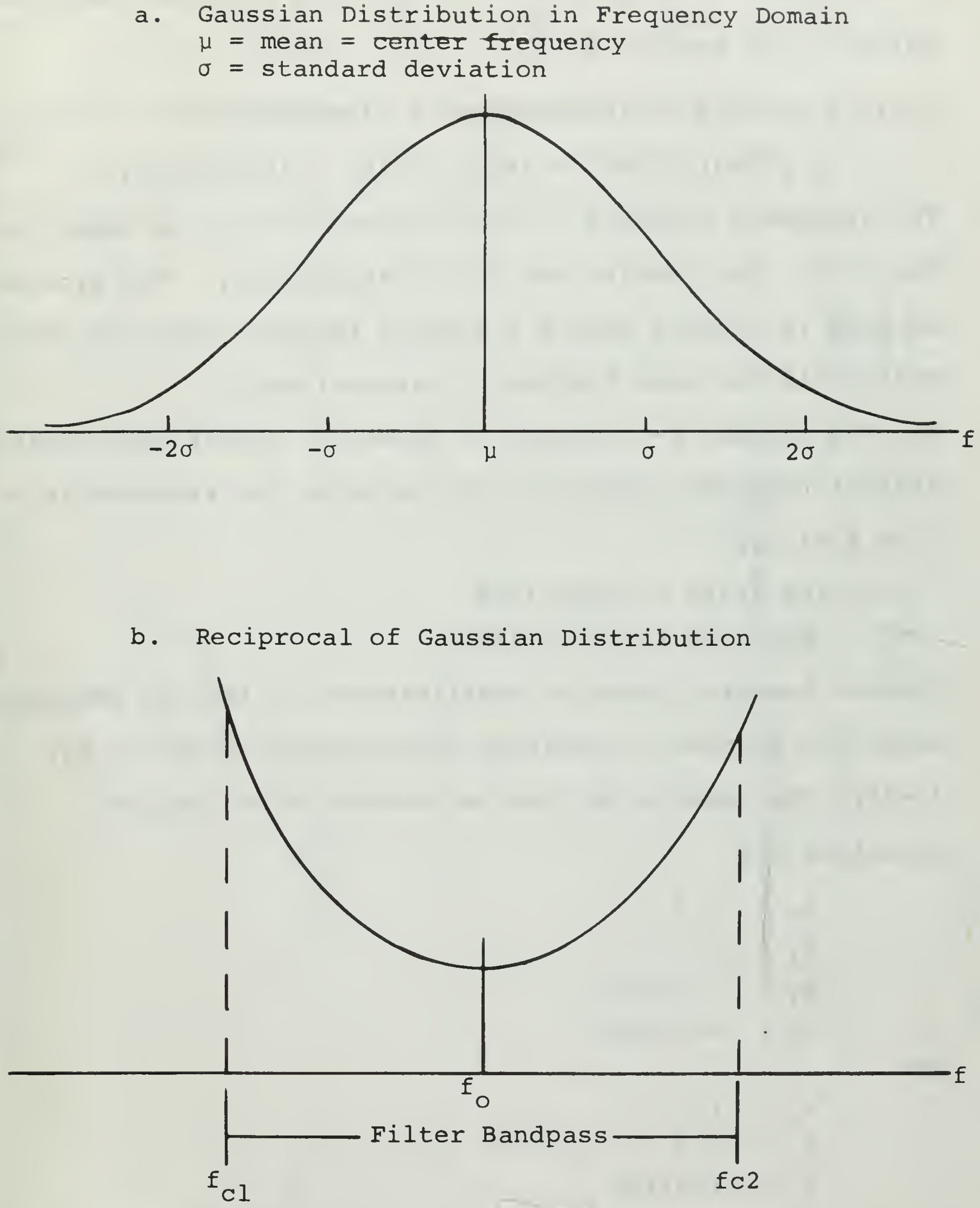

a. Gaussian Distribution in Frequency Domain
μ = mean = ~~center frequency~~
σ = standard deviation
-2σ
-σ
μ
σ
2σ
f
b. Reciprocal of Gaussian Distribution
f
$f_o$
Filter Bandpass
$f_{cl}$
fc2

Figure 3-5. A Noise Filter Specification.

designed using standard methods [10]. Finally a frequency transformation was performed to give the required bandpass filter. The result is

$$H(s) = 0.515422s^2/(0.000000052s^4 + 0.000039934s^3 + 1.250115878s^2 + 283.7773716s + 2610409.654). \tag{3-21}$$

The frequency response of this filter,[6] $|H(j\omega)|$, is shown in Fig. 3-6. The results are quite satisfactory. The problem at hand is then to obtain a digital transfer function which will yield the same frequency characteristic.

2. The matched z-transform is chosen to obtain the required digital transfer function. The poles of the transfer function H(s) are

$$s = -288.74512 \pm j4639.3984$$
$$s = -95.23544 \pm j1521.2546.$$

Digital transfer function coefficients can then be evaluated using the matched z-transform relationships given in Eq. (2-40). The results for the two second order transfer functions are

$$a_0 = 1.0$$
$$a_1 = -1.0$$
$$b_1 = 1.178165$$
$$b_2 = 0.749203$$

and

$$a_0 = 1.0$$
$$a_1 = -1.0$$
$$b_1 = -1381435$$
$$b_2 = 0.909159.$$

---

6 Figure 3-6 is the actual output from the Calahan network analysis computer program [11]. This program is an extremely useful aid in the present context of analog transfer function synthesis.

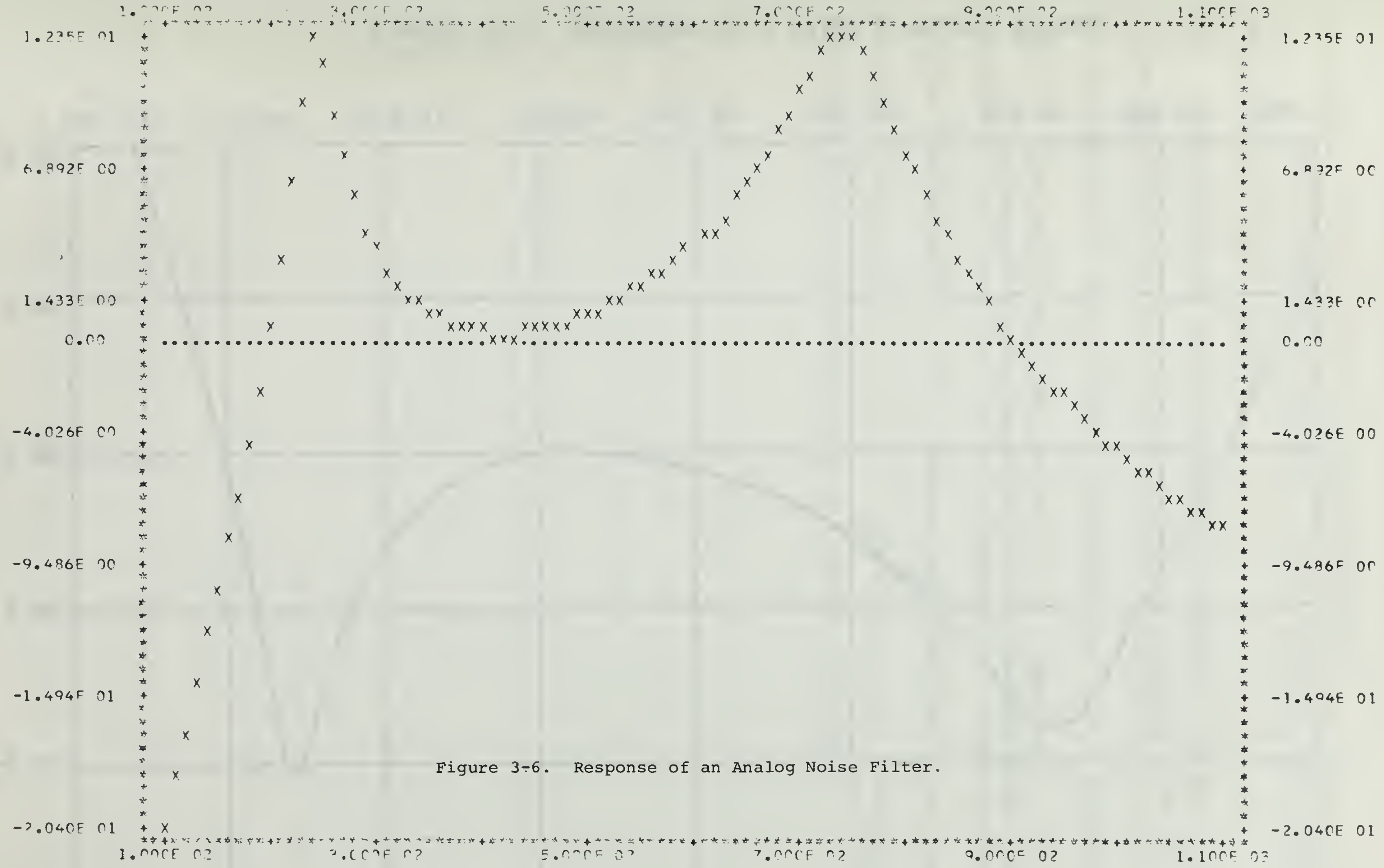


Figure 3-6. Response of an Analog Noise Filter.

```
MAX Y=  1.2351E 01 AT X=  7.3636E 02                MIN Y= -2.0404E 01 AT X=  1.0000E 02
MAX X=  1.1000E 03 AT Y= -8.1005E 00                MIN X=  1.0000E 02 AT Y= -2.0404E 01

PLOT OF ABOVE TABLE:  Y-AXIS  GAIN (DB)
                      X-AXIS  FREQ (HZ)
```

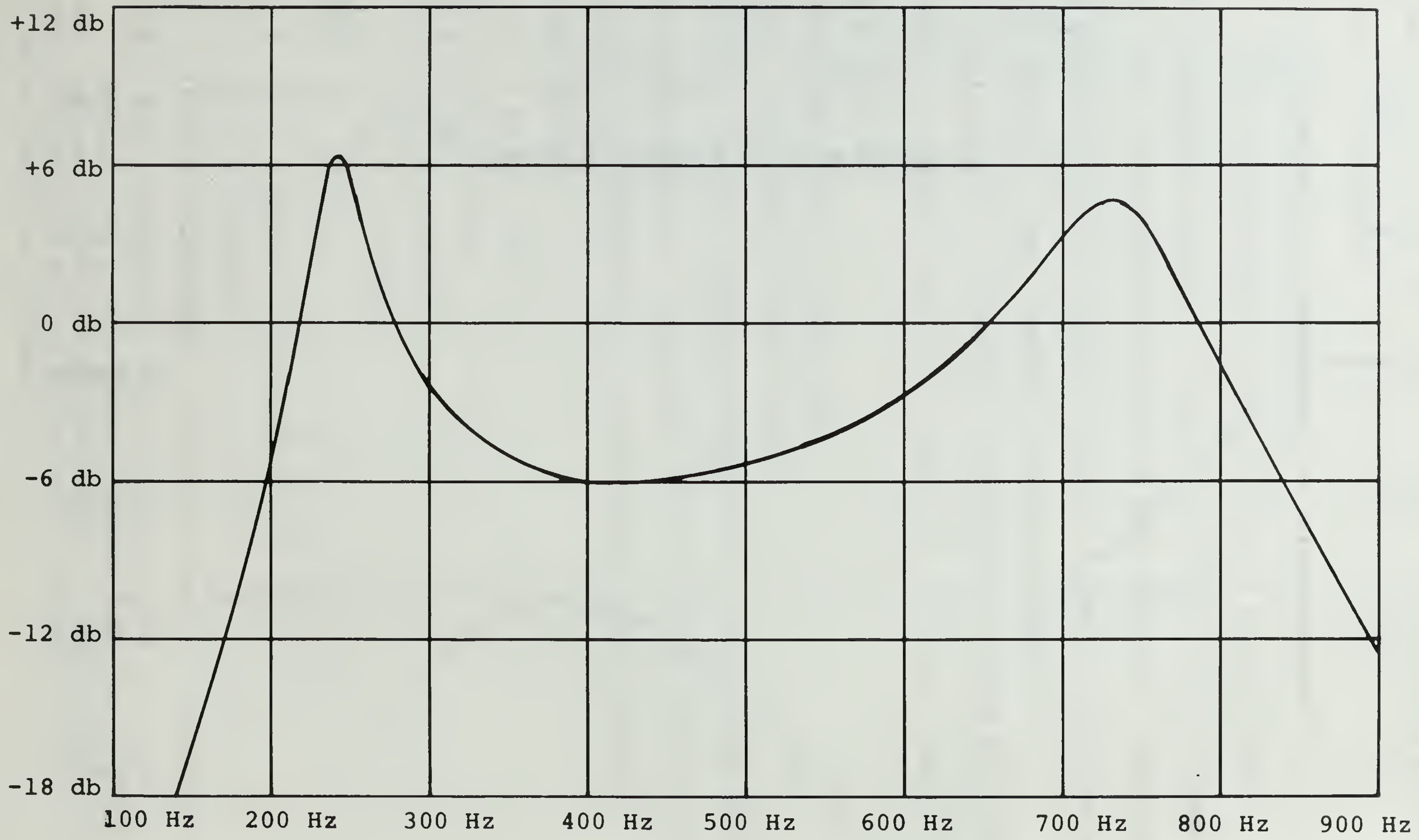


Figure 3-7. Response of a Digital Noise Filter.

Then the complete digital transfer function is

$$H(z) = \frac{K(1-z^{-1})(1-z^{-1})}{(1+1.178165z^{-1}+0.749203z^{-2})(1-1.381435z^{-1}+0.909159z^{-2})}, \qquad (3\text{-}22)$$

where K is a gain factor.

3. An examination of the frequency response of the transfer function given by Eq. (3-22) reveals a difference of 5.5 db in magnitude at the two peak frequencies. If a first-order zero at the half-sampling frequency is inserted in the transfer function, the desired symmetrical response can be better approximated. The response of this modified transfer function is plotted in Fig. 3-7 where the gain factor K was adjusted for 6-db attenuation at center frequency.

# IV. IMPLEMENTATION OF DIGITAL FILTERS

Let H(z) define an arbitrary digital transfer function derived by one of the methods discussed in Section III. The solution for the output transform, $Y(z) = H(z)X(z)$, can be written as

$$y(z) = \sum_{i=o}^{N} a_i \, z^{-i} \, X(z) - \sum_{i=1}^{M} b_i \, z^{-i} \, Y(z) \; ; \qquad (4\text{-}1)$$

or, in the time domain interpreting $z^{-1}$ as the unit delay operator,

$$y_n = \sum_{i=o}^{N} a_i \, x_{n-i} - \sum_{i=1}^{M} b_i \quad {}_{n-i} \; . \qquad (4\text{-}2)$$

This linear difference equation can be realized directly using digital components. Refer to Fig. 4-1. Each rectangle inscribed with $z^{-1}$ represents a one-sample delay. In real-time hardware this means a one-word memory whose capacity in bits is determined by the number of bits of accuracy used to represent $x_i$ and $y_i$. Also a multiplier is required for each branch in the filter. In general these are variable-coefficient multipliers to permit changing of the transfer function. Notice that these multiplying coefficients are the same coefficients which appear in the transfer function H(z). This is the basis for the earlier assertion that the design of digital filters is essentially complete once the coefficients have been specified. This is in sharp contrast to analog-filter design where the determination of the

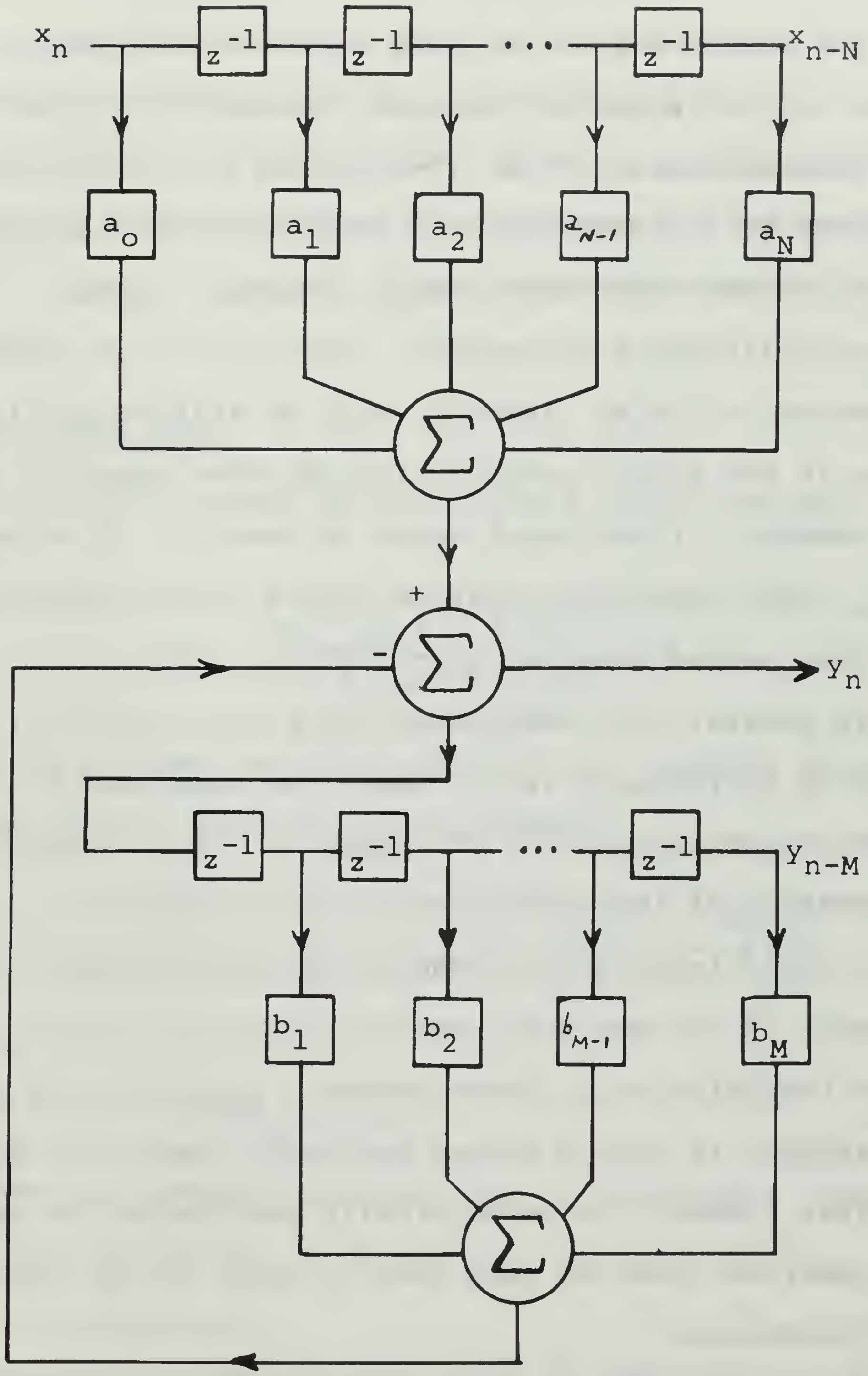


Figure 4-1. A Recursive Digital Filter.

transfer function H(s) is often only a small part of the design problem.

The adders and multipliers together comprise the arithmetic unit of a special-purpose computer or digital filter. The implementation of Eq. (4-2) would involve clocking this computer at a rate $f=1/T$, and performing the indicated calculations during each sample interval. After the calculations are complete, the contents of the storage registers, or delay elements, must be shifted one location to be in the proper position for the next sampling period. For example, if the input value at time $t = nT$ is stored as $x_n$, then this value must be stored in the location for $x_{n-1}$ one period later at $t = (nT+T)$.

In general, the expression for $H(z) = Y(z)/X(z)$ is a ratio of polynomials in $z^{-1}$ where the numerator is of degree N and the denominator is of degree M. If $N > M$ then the computation of the current value of the output $x_n$ in Eq. (4-2) would require knowledge of the input value $x_{n+1}$. For example, if N=5 and M=4, then the computation of $y_4$ would require the value of $x_5$ among others. Since this is physically impossible, it can be stated generally that for a digital transfer function to be physically realizable the degree of the numerator must be less than or equal to the degree of the denominator.

It is seen that the realization of digital filters involves basically a computational device to solve the linear difference equation which is defined by the digital transfer function of the filter. However, the sequence of performing the arithmetic operations and the method of storing intermediate answers do have an effect on the efficiency of the filter and the accuracy of the results obtained.[7] A digital filter is said to be in canonical form if, for a given order n, a minimum number of adders, multipliers and delays are utilized. Three canonical filter structures are now discussed.

## A. THE DIRECT FORM

Consider the general second-order digital transfer function

$$H(z) = \frac{a_o + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} . \qquad (4\text{-}3)$$

H(z) can be written as the product of two transfer functions, $H_1(z)$ and $H_2(z)$. Then Eq. (4-3) becomes

$$\frac{Y(z)}{X(z)} = \frac{1}{1 + b_1 z^{-1} + b_2 z^{-2}} \cdot \frac{a_o + a_1 z^{-1} + a_2 z^{-2}}{1} , \qquad (4\text{-}4)$$

and

$$Y(z) = X(z)H_1(z)H_2(z) . \qquad (4\text{-}5)$$

Define the intermediate variable $W(z) = X(z)H_1(z)$. Then, in the usual notation

---

[7] Efficiency refers to the speed of computation and the effective utilization of hardware. Both these items have become important "yardsticks" for measuring the usefulness of a digital filter.

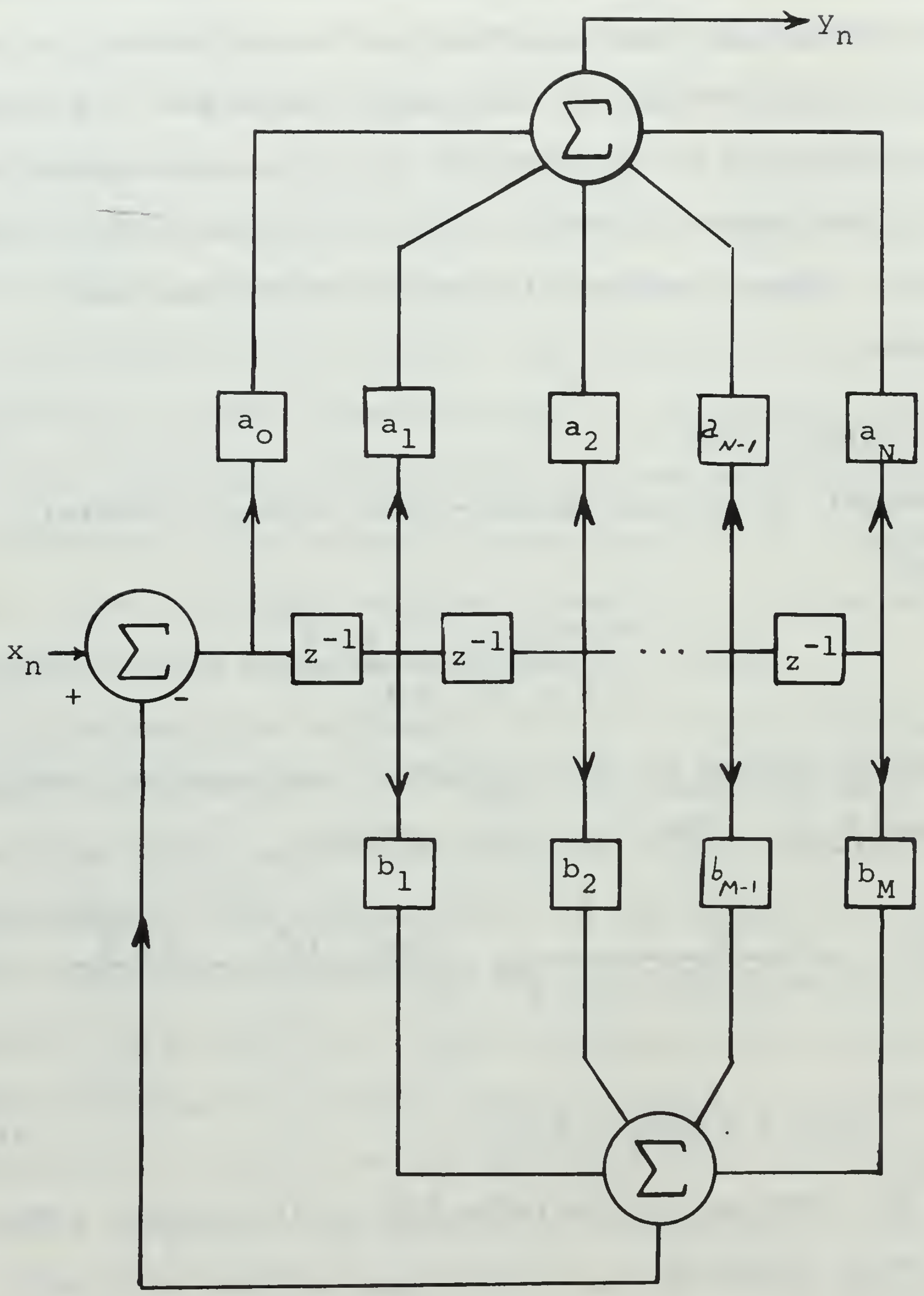


Figure 4.2. A Recursive Digital Filter in Canonical Form.

$$w_n = x_n - b_1 w_{n-1} - b_2 w_{n-2}$$
$$y_n = a_o w_n + a_1 w_{n-1} + a_2 w_{n-2} \quad (4\text{-}6)$$

The general result is that the single difference equation of the form of Eq. (4-2) is converted into two difference equations of the form

$$w_n = X_n - \sum_{i=1}^{M} b_i w_{n-i}$$
$$y_n = \sum_{i=o}^{N} a_i w_{n-i} \quad (4\text{-}7)$$

The schematic representation of Eq. (4-7) is shown in Fig. 4-2. Notice that Fig. 4-1 and Fig. 4-2 are similar in form but Fig. 4-1 contains redundant delay elements. Hence, Fig. 4-2 is a canonical filter structure. This means a more efficient use of hardware.

Any nth-order digital filter may be implemented in the forms of Figs. 4-1 and 4-2. These are both called direct forms and are the simplest of the recursive filter structures. However, because of numerical considerations, the direct forms are normally satisfactory only for low-order filters. These numerical considerations will be discussed in more detail in Section IV, D.

## B. THE CASCADE FORM

The general nth-order digital transfer function may be factored into a product of second-order transfer functions. The result is

$$H(z) = a_o \prod_{i=1}^{m} \frac{1 + \alpha_{1i} z^{-1} + \alpha_{2i} z^{-2}}{1 + \beta_{1i} z^{-1} + \beta_{2i} z^{-2}}, \qquad (4\text{-}8)$$

where m is the integer part of (n+1)/2. Note that for H(z) with poles and zeros in complex-conjugate pairs the coefficients $\alpha_i$ and $\beta_i$ will always be real. The resulting difference equations are of the form

$$\begin{aligned} Y_1(z) &= H_1(z)X(z) \\ Y_2(z) &= H_2(z)Y_1(z) \\ &\vdots \\ Y(z) &= H_m(z)Y_{m-1}(z) \end{aligned} \qquad (4\text{-}9)$$

by the same arguments which were made in the preceding section.

The schematic representation of Eq. (4-9) is shown in Fig. 4-3. It is seen to be a cascade of second-order subfilters and this structure has several important features. First the structure can easily be multiplexed. By using a basic second-order filter structure and appropriate timing and control circuitry, a high-order filter can be implemented by time-sharing the second-order structure. The appropriate coefficients, $\alpha_i$ and $\beta_i$, are supplied for the m subfilters in time sequence.

Also notice that if H(z) contains zeros on the unit circle, the coefficient $\alpha_{2i}$ in the subfilter which contains a second-order zero will be unity. This means that one less multiplication is required. Such a situation occurs frequently when the bilinear z-transform is used to design bandpass and bandstop filters.

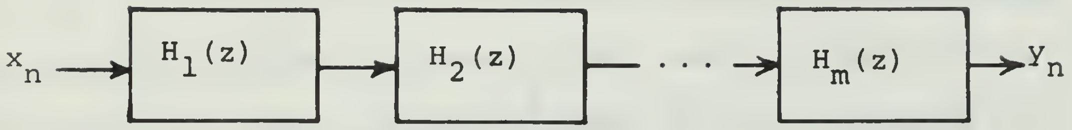


Figure 4-3. The Cascade Form

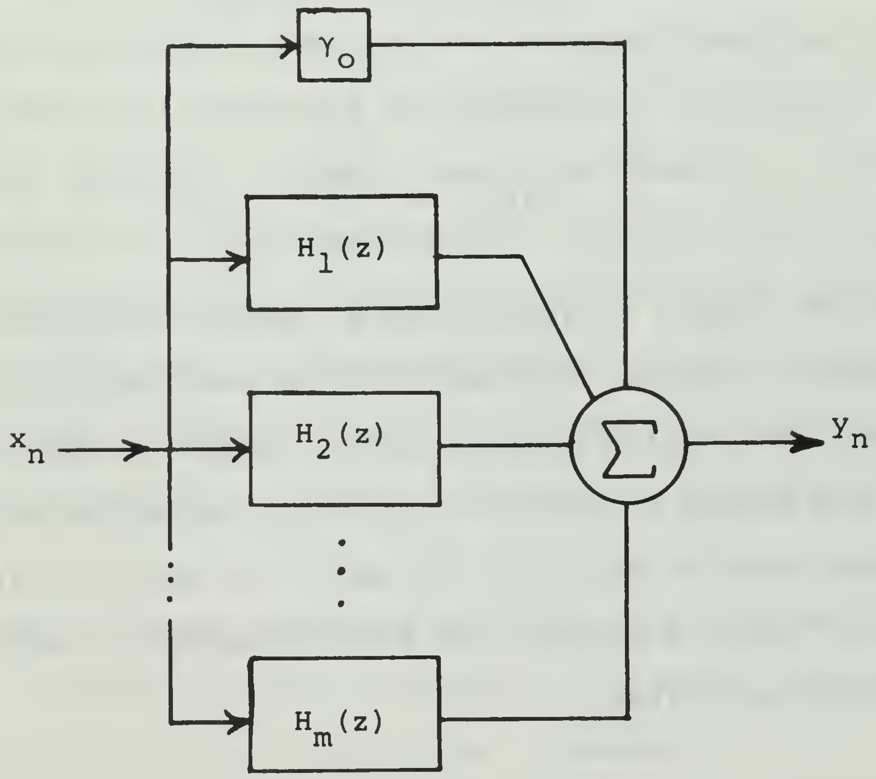


Figure 4-4. The Parallel Form

## C. THE PARALLEL FORM

The third canonical form of a digital filter is obtained by expanding H(z) in partial fractions of second-order terms. The result is

$$H(z) = \gamma_o + \sum_{i=1}^{m} \frac{\gamma_{oi} + \gamma_{1i} z^{-1}}{1 + \beta_{1i}z^{-1} + \beta_{2i}z^{-2}} , \qquad (4\text{-}10)$$

where $\gamma_o = a_n/b_n$.[8] Equation (4-10) yields difference equations of the form

$$\begin{aligned} Y_o(z) &= \gamma_o X(z) \\ Y_1(z) &= H_1(z)X(z) \\ &\vdots \\ Y_m(z) &= H_m(z)X(z) \\ Y(z) &= \sum_{i=o}^{m} Y_i(z) ; \end{aligned} \qquad (4\text{-}11)$$

and the filter is structured as shown in Fig. 4-4. The parallel form is comparable to the cascade form in performance. The choice between the two might be made on the basis of the amount of algebra involved. Depending on the z-transformation employed the amount of work in arriving at a final filter structure can be considerable, especially for high-order filters.

[8]Alternatively the expansion may be made of H(z) written as a function of z rather than $z^{-1}$.

## D. NUMERICAL CONSIDERATIONS

Until this point it has been assumed that both the calculation of digital filter coefficients and the manipulation of these coefficients in the filters themselves were done with infinite accuracy. Since this cannot be the case, and since even the input samples to the digital filter are only an approximation to the instantaneous values of the analog signal, the actual results may differ markedly from the theoretical results.

As with any numerical process the accuracy of the calculated "answers" using digital filters is a very critical function of the hardware used and the computational technique employed. The finite computer word length is the basic limitation on accuracy. Errors introduced because of this limitation are: (1) finite precision in filter coefficients; (2) quantization of input data to a specific number of bits; and (3) round-off error in multiplications and additions. References 1 and 2 contain rigorous treatments of quantization effects in digital filters. The results are summarized below.

Numerical problems can begin even with the digital filter design. The design methods of Section III involve factorization and partial fraction expansions. Generally it is recommended to perform these operations in the s domain and to digitalize the resulting subfilters. Although factoring of the polynomials in s is not required for the bilinear z-transform, better results are obtained if this is done. The generation and manipulation of high-order polynomials in z should be avoided. In particular, finding roots of polynomials in z can be quite difficult because of the tight clustering of poles and zeros in the z-plane.

The popular notion that a higher sampling frequency always produces better results is not true. Actually a higher sampling rate requires greater accuracy of computation. Performance may deteriorate if the sampling rate is increased beyond certain limits.

Often the most critical numerical problem in digital filters is the quantization of the filter coefficients. These coefficients determine the locations of the poles and zeros for the filter and any perturbation in the coefficients causes a shift in the pole and zero locations. For high-order filters the poles are usually clustered in a small area near the unit circle in the z-plane. Perturbations in these coefficients may actually cause the filter to become unstable. Furthermore, the pole locations are much more sensitive to coefficient perturbations when using the direct filter structure. Consequently high-order filters are almost always realized in the cascade or parallel form. Weaver et al [7] present a procedure for determining the bounds on the sensitivity of pole positions to coefficient perturbation. Given these bounds the required coefficient accuracy can be calculated.

The quantization of the input signal in the analog-to-digital converter is treated as a noise input to the filter.

The round-off error in the arithmetic unit is also treated as noise but its entry into the filter may occur in various places depending on the realization structure. In the direct and parallel forms the quantization noise due to round-off error is simply additive. However in the cascade form round-off error may cause unsatisfactory operation because of the multiplicative effect in a serial system. A complete analysis of various implementation schemes can be accomplished using noise models such as those developed by Gold and Pader [4].

## E. HARDWARE CONSIDERATIONS

Any device which operates on signals in real-time according to a digital transfer function is, in essence, a special-purpose digital computer. The computer program, or equivalently the transfer function and implementation scheme, can be hardwired directly into the computer. The input data would then be the signal samples and the filter coefficients. It is anticipated that LSI will provide the mechanism for incorporating such a computer into a package that is small in size, economical in cost, and reliable in performance. The design of such a package is a formidable task, and several of the more important considerations are now discussed.

The first consideration is the choice of the sampling or clock frequency. Unfortunately the requirements of a digital filter in this respect are self-opposing. To obtain sharp cutoffs, more narrow bandwidths, deeper notches, etc., higher-order filter functions are required with correspondingly large computation times. Since the computation must be accomplished in a single sampling interval, the sampling frequency must go down to provide a longer time T. However, as the sampling frequency is lowered, the usable frequency spectrum is reduced because the relation $\omega_c < \omega_s/2$ must be maintained.

The second major consideration is computer word length. The cost and complexity of a real-time digital filter varies in direct proportion to the number of bits which must be stored and manipulated. It appears that the key to success in reducing computer word length lies in developing new implementation schemes which are less sensitive to numerical inaccuracies.

In general, therefore, it is of the utmost importance to design and organize special-purpose hardware which can perform digital-filter type computations efficiently, accurately, and as fast as possible. One of the more demanding computations in a digital filter is multiplication. In order to illustrate some of the problems in logical design peculiar to digital filters a binary serial multiplier is outlined in detail in Appendix A. The multiplier follows the general design proposed by Jackson et al in Ref. 8.

## F. THE FAST FOURIER TRANSFORM

The methods discussed thus far in implementing digital filters all involve the technique of interpreting $z^{-1}$ as the unit delay operator and obtaining a solution of linear difference equations directly in the time domain. This technique is perfectly valid for analyzing the steady-state response of digital transfer functions to sinusoidal inputs, and has been shown to be very practical in real-time applications. However, for complex filters, and in particular for non-recursive filters with a large number of terms, the number of multiplications and additions may be prohibitive. In these cases filter implementation by high-speed convolution using the fast Fourier transform (FFT) becomes desirable.

Recall that the discrete convolution of an input sequence with a filter impulse response is given by

$$y_n = \sum_{i=o}^{M-1} h_n x_{n-i} , \qquad (4\text{-}12)$$

where M is the number of samples being convolved. When M is large evaluation of Eq. (4-12) by z-transform methods becomes difficult and time consuming.

Alternatively the sequence $y_n$ may be evaluated by (1) taking the Discrete Fourier Transform (DFT) of the input sequence and the impulse response of the filter, (2) multiplying these two transforms together, and (3) taking the inverse DFT of the result. This procedure is very similar to using the conventional Fourier transform to analyze continuous-time signals in the frequency domain. However the theory of the DFT is somewhat more involved.[9]

---

[9] This is due to the fact that the product of two DFT's in the frequency domain is equivalent to circular convolution in time, not linear convolution as intended in Eq. (4-12). Nevertheless, the general results as stated above are correct.

The practicality of this method lies in the ability to use the FFT techniques to evaluate the DFT's. The fast Fourier transform is described in detail in Refs. 4 and 12. Briefly it makes use of the fact that under certain conditions the calculation of the coefficients can be done iteratively with a considerable saving in time. Generally the value M (the number of sample values) must be a highly composite number (contains many factors). When M is a power of 2 the FFT requires a number of computations proportional to $M\log_2 M$ vice $M^2$ for the conventional convolution.

Presently a large general-purpose computer is required to perform high-speed convolution via the FFT. It is anticipated that this algorithm will eventually be programmed into special-purpose hardware for real-time applications.

# V. REAL-TIME SIMULATION OF DIGITAL FILTERS

In recent years large general-purpose computers have been used extensively to perform the computations required in digital signal processors. Computer Program 1 is an example of the frequency-domain analysis of a digital transfer function. In the time domain the calculation of an output sequence of a digital filter can also be programmed for a general-purpose computer. In this case the actual difference equations are solved and the program becomes a simulation of the digital filter. One limitation in such a scheme is immediately evident; a large computer memory is required to store the input and output number sequences. Nevertheless, such computer simulations of digital filters do find wide application. Indeed, most algorithms involving the use of the fast Fourier transform are performed on a general-purpose computer.

With the increased emphasis on the real-time implementation of digital filters it becomes desirable to be able to simulate digital filters in real time. Such a simulation would provide an experimental basis for establishing specifications for the realization of digital filters in special-purpose hardware. It would provide a means for investigating and verifying new design and implementation techniques. And finally it would provide a real signal to observe, measure, and analyze both quantitatively and qualitatively.

A real-time simulation of digital filters must contain certain features if it is to be used as an experimental analysis and design tool. First it must be possible to control the two basic design parameters--clock frequency and computer word length. Precise control of the sequence of arithmetic operations is required to simulate different filter structures. And finally the simulation scheme must contain the necessary input-output interfaces to permit the complete processing of a signal waveform in real time. In short, it is required that a simulation scheme be developed which can duplicate as closely as possible an arbitrarily designed special-purpose computer or digital filter.

A real-time digital-filter simulation in the above context cannot be accomplished using only a large general-purpose computer. Aside from the obvious lack of any analog signal interface, the control of computer word length and precise arithmetic sequence is difficult, if not impossible, to achieve in most large-computer operational environments. Furthermore, the use of a large computer for relatively simple computational algorithms for an extended period of time is extremely inefficient and, hence, undesirable.

On the other hand, the real-time simulation problem is well suited to a scientifically oriented hybrid-computer system. Some of the characteristic features of such a system are:

1. Built-in interface between analog and digital computers including analog-to-digital converters and digital-to-analog converters.

2. Readily accessible analog functions such as amplification, scaling, clipping, integration, etc.

3. Readily accessible digital functions such as clocking, counting, storing, calculating, programming, etc.
4. Accessibility of such peripheral equipment as oscilloscopes, vacuum-tube voltmeters, signal and function generators, line printers, card readers, tape recorders, and teletypewriters.
5. Typical "hands on" operation of the computer system vice the "batch processing" type operation of a computer facility.

In the following sections a real-time simulation for digital filters is presented. The specific simulation scheme was designed for use on the hybrid-computer system located in the Department of Electrical Engineering at the Naval Postgraduate School. While the ensuing discussion must be specific in certain areas, the solution to the overall problem is quite general.

## A. THE HYBRID-COMPUTER CONFIGURATION

### 1. <u>The Physical System</u>

The hybrid-computer system at the Naval Postgraduate School consists of the following major subsystems:

1. Xerox Data Systems XDS 9300 Computer[10]
2. COMCOR Ci-5000 Analog Computer
3. Hybrid interface subsystem consisting of:
    a. Adage VMX multiplexer
    b. Adage VT-13 14-bit A/D converter

Some of the characteristics of the XDS 9300 digital computer which are of specific interest are:

1. 24-bit word including sign bit
2. META-SYMBOL Symbolic Assembler

[10] This nomenclature was formerly "Scientific Data Systems SDS 9300 Computer."

3. 1.75-microsecond cycle time
4. 1.75-microsecond fixed-point add time
5. 7.0-microsecond fixed-point multiply time
6. direct memory access for A/D converter output
7. integral 15-bit D/A converter
8. 3 index registers available to the programmer
9. priority interrupt system

The Ci-5000 analog computer has a full compliment of operational amplifiers, potentiometers, passive analog components, logic components, clock sources, and computer control devices. These items will be explained as necessary in the discussion.

The Adage A/D converter produces a 14-bit binary fraction including sign bit from an analog input signal within the range $\pm$ 100 volts. This 14-bit fraction is stored in the most significant (leftmost) portion of an XDS 9300 24-bit memory location. That is,

$$+100V = 37776000_8$$
$$0\ V = 00000000_8$$
$$-100V = 40000000_8$$

where the decimal point is assumed after the sign bit, and negative values are stored in two's complement form.

## 2. The General Simulation Scheme

The digital-filter simulation which follows is built around a digital-computer program written in a symbolic programming language. This symbolic language, called META-SYMBOL, is the means whereby a machine-language program is generated in the XDS 9300 computer.

A machine-language program is desirable for several reasons. First it provides the rigid computer control and accessibility that is required for real-time filter simulation. For example, arithmetic operations such as addition and multiplication are specified precisely in the desired sequence. The location in memory of each variable, machine instruction and data word is known, and the flow of digital information in the computer is completely specified by the program. Furthermore, a machine-language program permits the operator to monitor the execution, enter the program to insert modifications, or to stop the program to examine intermediate results. In general, the programmer not only tells the computer what to do, but how, where, and when to do it. This complete control of the situation is required for the accurate simulation of a special-purpose computer.

The analog signal which is to be filtered is fed to the patchboard of the analog computer. It is amplified, if necessary, and then fed directly to a trunk line that is connected to one of the input channels of the A/D converter. Operation of the A/D converter is under control of the digital-computer program. At a certain point in the program the analog signal is sampled and the quantized signal magnitude is stored directly in a previously specified location in memory. The digital computer then begins to perform the arithmetic operations of the digital filter. The filter coefficients had previously been stored in memory and the sequency of operations is programmed to follow the filter

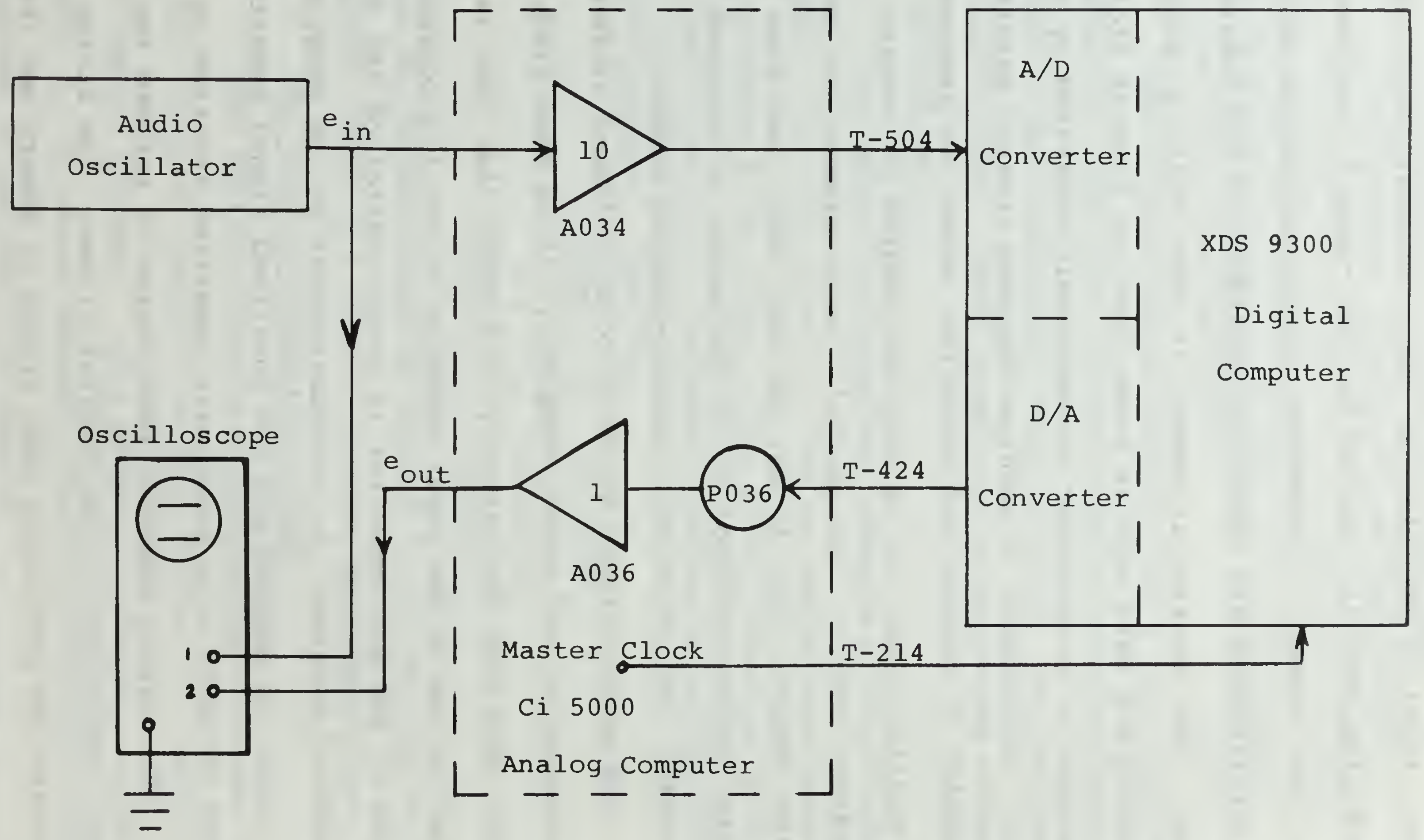


Figure 5-1. Block Diagram of the Hybrid-Computer Configuration

implementation scheme exactly. Upon completion of the calculations the output value for that sample period is stored and also made available to the D/A converter. When the D/A conversion is complete the filtered analog signal is fed back to the analog patchboard where it is rescaled and applied to an oscilloscope or spectrum analyzer. Back in the digital computer the various input and output sample values which are used for computation are shifted in memory to be in the proper location for the next sample period. The computer then waits for a clock pulse to begin the next iteration of the entire sequence.

The master clock is located in the analog computer. Each clock pulse causes an interrupt in the digital computer operation and immediately causes it to go to the starting point in the iterative program. Obviously the digital computer must have completed the calculations for a given sampling period before the next clock impulse occurs. Thus the computation time of the digital filter sets an upper limit on clock frequency.

### 3. The Hybrid-Computer Diagram

A block diagram of the simulation scheme is given in Fig. 5-1. An audio signal generator is used as a signal source. Because of the $\pm$100V input range in the A/D converter the original signal is scaled by a factor of 10 in amplifier A034. This means the input signal must now be limited to $\pm$ 10.0V. The unscaled signal, $e_{in}$, is also fed to the upper trace on the oscilloscope.

The output of the D/A converter, which is the filtered signal, is fed to a unity-gain amplifier A036 through potentiometer P036. Assuming no additional scaling of $e_{in}$ was required in the digital computer a setting of 0.1 on P036 should cancel the gain factor of 10 on the input side, and the magnitude of $e_{out}$ should be correct according to the transfer function of the filter. If a given value of $e_{in}$ produces overflow in the computer's arithmetic unit, the overflow indicator on the console is lit. In this case the signal input level can simply be reduced, or additional scaling (usually $2^{-1}$ or $2^{-2}$) can easily be accomplished in the computer program. Potentiometer P036 is then adjusted to equate the net scale factor of the entire loop to unity.

## B. THE DIGITAL-FILTER SIMULATION PROGRAM

Computer Program 2 is a general program which can be used for the real-time simulation of digital filters on the hybrid-computer system discussed in the preceding section. The source program is written in the META-SYMBOL language and appears on the right side of the page. The block of numbers on the left side of the page is the corresponding machine-language program generated by the META-SYMBOL assembler in the XDS 9300 computer. A complete description of the operation of this computer and the mechanics of the META-SYMBOL language will not be given here. Detailed information on both subjects is contained in Refs. 13 and 14 respectively. However, the following limited information is provided in order to explain the simulation program.

## 1. Background Information on the XDS 9300 Computer

The META-SYMBOL source program is originally punched on cards. Refer to Computer Program 2. Card column 1 is printed in the location of the asterisk in line number 1. The format for the symbolic program is as follows:

1. Card columns 1-7 contain the label field. This field is used for address identification or labeling.
2. Card column 8-15 contain the operation field. Either mnemonic machine instructions or assembly directions are entered in this field.
3. Card columns 16-30 contain the operand field and may be used for various purposes. Put very simply, the operand answers the questions "what?" or "where?" regarding the operation field.
4. Card columns 31-80 contain the comments field and are ignored by the computer.

The computer word length in the XDS 9300 is 24 bits or eight octal digits. The rightmost eight digits in the block of numbers in the program printout represent the machine instruction corresponding to the META-SYMBOL instruction on the same line. Each machine instruction and data word in the program is stored in the memory location given by the first five digits in that line. For example, refer to line 34 in the program. The machine instruction 01600132 is stored in memory location 00035. Furthermore, the eight octal digits of the instruction word are formatted as follows:

Digit 1 contains address and index register information.
Digits 2 and 3 contain the instruction code.
Digits 4 through 8 contain the address of the operand.

An example will help clarify the discussion. Again referring to line number 34 in the program, the META-SYMBOL statement reads, "LDA ••• SUM+1." This means simply, "load the A register with the contents of memory location SUM+1." The operation code for LDA is 16. Therefore, the machine instruction states, "load the A register with the contents of memory location 00132." From line number 100 it is seen that the location in memory which is one greater than the location reserved for SUM is in fact location number 00132.

Address modification using indexing and indirect addressing is somewhat more involved and will not be explained here. Moreover, many of the instructions in the program are required simply to make the program work and do not warrant explanation beyond the comments in the printout itself. Most of the "subroutines" (ADOPS, etc.) are required by the hybrid interface software.

## 2. Background Information on the META-SYMBOL Language

The most important part of the simulation program involves the use of the computer's arithmetic unit to perform the prescribed calculations on certain sample values. The following list contains selected machine instructions from the META-SYMBOL program language which are used to simulate digital filters.

| Instruction | Translation |
|---|---|
| LDA | Load the A register with the contents of the effective memory location. |
| STA | Store the contents of the A register in the effective memory location. |
| ADM | Add the contents of the A register to the effective memory location. |

| | |
|---|---|
| MUL | Multiply the contents of the A register by the contents of the effective memory location. The full product is contained in the A and B registers. |
| ALSD | The contents of the A and B register, treated as a double-length register, are shifted left a specified number of bits. |
| ARSA | A right shift similar to that above involving the A register only. |
| BRM | Mark Place and branch. |
| BRU | Branch unconditionally. |
| LDX | Load the specified index register with a specified number. |
| BRX | Branch only if the associated index register has not reached its terminal state. |

The general use of the above instructions in the digital simulation are:

1. LDA, STA, MUL, AND ADM are used to add or multiply and to store results in a desired location.
2. ALSD is equivalent to multiplying by $2^n$ where n is the number of bits shifted.
3. ARSA is equivalent to dividing by $2^n$.
4. BRM is used to shift computer control to a desired "subroutine." The return to the program is automatic when execution of the subroutine is complete.
5. BRU is equivalent to a FORTRAN GO TO statement and is used in this program to iterate a sequence of instructions indefinitely.
6. LDX and BRX provide a "DO LOOP" mechanism whereby a certain group of instructions is performed a prescribed number of times. Indirect addressing is often used to shift an effective address for each new iteration.

### 3. Fixed-Point Arithmetic

The XDS 9300 computer performs either floating-point arithmetic or integer arithmetic. Since floating-point arithmetic is far more complicated, and since it would almost never be used in special-purpose hardware, integer arithmetic is the most practical alternative. However, in the digital filter itself, fixed-point arithmetic is desired whereby a mentally placed decimal point in a computer word remains fixed throughout the calculations. Hence, a conversion scheme is required.

Recall that the basic word length in the XDS 9300 computer is 24 bits including sign. The output of the A/D converter is a 14-bit word corresponding to a binary fraction of magnitude less than one. That is, the first bit is a sign bit and bits 2-14 are magnitude bits assuming a decimal point after the sign bit. The least significant 10 bits of a sample value in memory, therefore, are all zero. The word format may be written symbolically as

X.XXXXXXXXXXXXX0000000000

where X indicates a one or zero. Since the magnitude of the input sample is less than one, this word can always be treated as a 23-bit integer (excluding the sign bit) in the computer, and the decimal point may be inserted mentally whenever it is desired to determine the real value. For example, if the fraction $(0.5)_{10} = (0.01)_2$ is added to the fraction $(0.25)_{10} = (0.01)_2$, then the computer performs the addition

$$\begin{array}{r} 010000000000000000000000 \\ \underline{001000000000000000000000} \\ 011000000000000000000000 \end{array}$$

The answer is $(11\text{x}2^{20})_2 = (6291456)_{10}$. However if a decimal point is mentally reinserted after the sign bit, the correct answer is seen to be $(0.110\cdots0)_2 = (0.75)_{10}$. If the 24-bit word in the answer above is input to the D/A converter, it correctly interprets the value as 0.75 and the output is 75V. Thus, no conversion is necessary for data which is input via the A/D converter.

The situation is quite different for the multiplying coefficients, $a_i$ and $b_i$. If a coefficient of $(0.5)_{10}$ were to be entered into the program as data, the integer ($\text{base}_2$) equivalent of $(0.5)_{10}$, which is zero, would be stored in memory. This is because the assembler converts all base 10 data to base 2 integers when using integer arithmetic. The solution is to multiply $(0.5)_{10}$ by $2^{23}$. If this were done, then the 23-bit integer that is generated in the computer will be $2^{23}$ times too large which is exactly what is desired.

$$(0.5x2^{23})_{10} = (4194304)_{10} = (010000000000000000000000)_2$$

Hence, if the base 10 integer 4194304 is entered as data the desired fraction is stored in memory.

Since the $a_i$ and $b_i$ for most cascade and parallel filter forms are within the range -2 to +2, a more convenient conversion is

$$C_i = c_i x 2^{22} = c_i(4194304), \qquad (5\text{-}1)$$

where the general notation $c_i$ includes all multiplying coefficients. The conversion in Eq. (5-1) is equivalent to dividing the coefficient by two to preclude any overflow in the A register. After the multiplication is complete the answer is multiplied by two to obtain the correct numerical result. In Eq. (5-1) $C_i$ is the base 10 integer entered in the computer program.

4. A Discussion of the Program

Computer Program 2 contains the scaled coefficients for the second-order low-pass filter which was designed in Section III, C. The digital transfer function for that filter is

$$H(z) = \frac{0.067455 + 0.134910z^{-1} + 0.067455z^{-2}}{1 - 1.142980z^{-1} + 0.412801z^{-2}}, \qquad (5\text{-}2)$$

and the corresponding difference equation is

$$y_n = a_o x_n + a_1 x_{n-1} + a_2 x_{n-2} - b_1 y_{n-1} - b_2 y_{n-2}. \qquad (5\text{-}3)$$

If the coefficients are scaled according to Eq. (5-1), the result is

| | |
|---|---|
| $a_0 = 0.067455$ | $A_0 = 282927$ |
| $a_1 = 0.134910$ | $A_1 = 565854$ |
| $a_2 = 0.067455$ | $A_2 = 282927$ |
| $b_1 = -1.142980$ | $B_1 = 4794006$ |
| $b_2 = 0.412801$ | $B_2 = -1731413$, |

where the signs of the $B_i$ are reversed to permit addition instead of subtraction.[11]

The sample values are stored in memory in the following order

| | |
|---|---|
| DATA | $= x_n$ |
| DATA+1 | $= x_{n-1}$ |
| DATA+2 | $= x_{n-2}$ |
| SUM | $= y_n$ |
| SUM+1 | $= y_{n-1}$ |
| SUM+2 | $= y_{n-2}$. |

[11] The reversing of the signs of the $B_i$ in the program applies to all implementation schemes and will be done for all remaining examples without comment.

Then the sequence of instructions from line 22 to line 41 in the program solves the following equation:

$$\text{SUM} = \frac{\text{A0}}{2} \cdot (\text{DATA}) \cdot 2 + \frac{\text{A1}}{2} \cdot (\text{DATA+1}) \cdot 2 + \frac{\text{A2}}{2} \cdot (\text{DATA+2}) \cdot 2 + \frac{\text{B1}}{2} \cdot (\text{SUM+1}) \cdot 2 + \frac{\text{B2}}{2} \cdot (\text{SUM+2}) \cdot 2 \ , \tag{5-4}$$

which is equivalent to Eq. (5-3). Thus, Computer Program 2 is a simulation of the first of the two direct filter structures discussed in Section IV, A.

Notice that all the coefficients were divided by two to permit a more general program. Also recall that, because integer arithmetic is being used, the most significant portion of a product is contained in the A register. (The entire product is in the A and B 48-bit register.) If the contents of the B register are dropped completely, a correct 24-bit product is still retained since the value is interpreted as a binary fraction.

At the end of the arithmetic sequence the variables are shifted in memory to prepare for the next sampling interval. This is accomplished by the instructions in lines 43 through 46. This sequence is performed five times (until the contents of the index register become negative), and the effective addresses of the load and store instructions are decremented by one for each iteration. The resultant shift in memory is shown below:

| Shift | Order of Occurrence |
|---|---|
| DATA | last |
| ↓ | |
| DATA+1 | 5th |
| ↓ | |
| DATA+2 | 4th |
| ↓ | |
| SUM | 3rd |
| ↓ | |
| SUM+1 | 2nd |
| ↓ | |
| SUM+2 | 1st |

The current DATA+2 and SUM+2 values are lost which is of no consequence. The contents of SUM become correct in the next sample period after the STA instruction is used in line 25.

Implementation of more complicated filters will be illustrated in the examples which follow. The only differences in the program are the sequence of arithmetic operations and the arrangement of the data and the coefficients in memory.

In the beginning of this section on digital-filter simulation it was mentioned that it would be desirable to control the computer word length. This can be done conveniently in the program by the use of the ETR (extract) instruction. Refer to line 64 of Computer Program 2. The ETR instruction performs a logical AND operation between the contents of the A register and the operand of the ETR instruction. In the D/A subroutine the output value $y_n$ is loaded into the A register. Then a logical AND is performed between $y_n$ and the contents of MASK which are

$$(111111111111111000000000)_2$$

The result is that $y_n$ is truncated to 15 bits which is the word size used in the D/A converter. If an ETR instruction is inserted after each multiplication, then all the arithmetic in the digital filter is truncated to the number of bits specified in location MASK. This means that the effective word length of the computer can be controlled by altering the contents of location MASK. Additionally the accuracy of the multiplying coefficients can be reduced by truncating the number of significant bits in the storage locations. This would normally be done manually since the coefficients are constant in time.

# VI. EXPERIMENTAL RESULTS OF REAL-TIME DIGITAL-FILTER SIMULATIONS

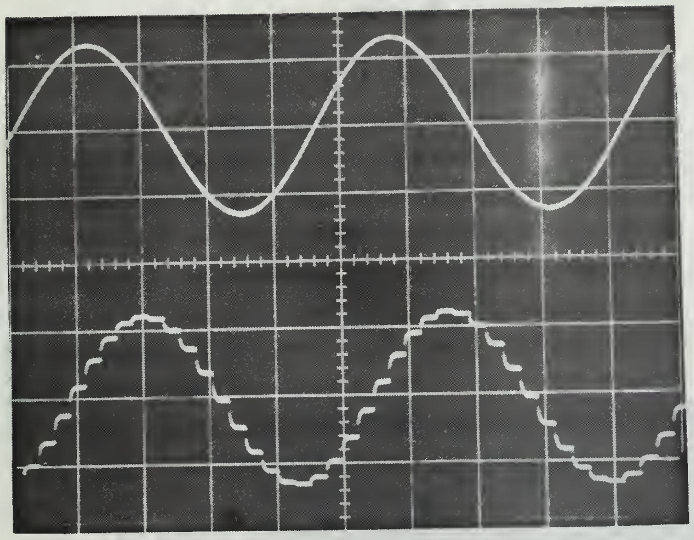
The digital-filter simulation scheme described in the previous section was used successfully for the real-time simulation of various digital filters. The same basic program may be used for all filters. Different filter structures are simulated by changing the sequence of arithmetic operations. Different filters of the same structure may be simulated by changing the filter coefficients in the program. For the remaining examples only the arithmetic and data-shifting sequences and the filter coefficients are shown in the referenced computer programs.

## A. A LOW-PASS FILTER

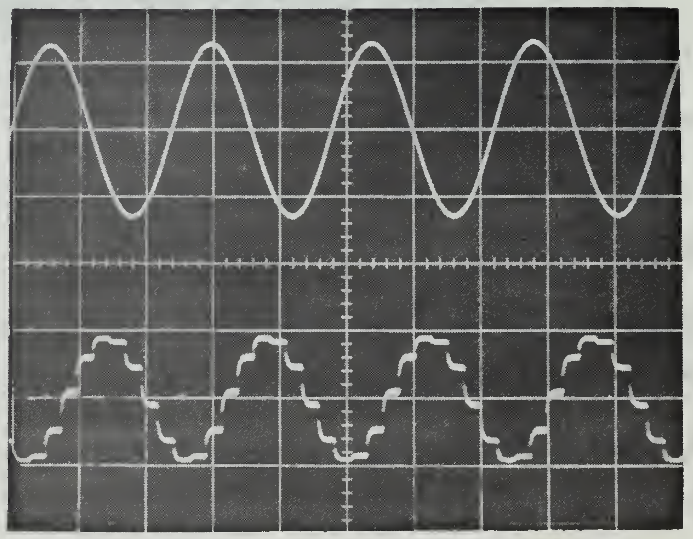
A second-order low-pass filter with a cutoff frequency equal to 200 Hz and a sampling frequency equal to 2000 Hz was designed in Section III, C. The transfer function of that filter is

$$H(z) = \frac{0.067455 + 0.134910z^{-1} + 0.067455z^{-2}}{1 - 1.142980z^{-1} + 0.412801z^{-2}} . \qquad (6\text{-}1)$$

This is the same filter that was used to describe the general simulation program. Computer Program 3 shows the arithmetic and data-shifting instructions necessary to implement this filter in the second-order canonical form. Notice that in this case additional scaling of the input data became necessary. This was easily accomplished using the ARSA instruction in line 23. The difference equations which are



$f_s$ = 100 Hz
Time = 2 msec/div
Amplitude = 5V/div



$f_s$ = 200 Hz
Time = 2 msec/div
Amplitude = 5V/div

Figure 6-1. An Output Signal from a Low-Pass Digital Filter.

implemented by Computer Program 3 are

$$\frac{SUM}{4} = \frac{SUM}{4} + \frac{B1}{2}\cdot\left(\frac{SUM+1}{4}\right)\cdot 2 + \frac{B2}{2}\cdot\left(\frac{SUM+2}{4}\right) 2 \qquad (6\text{-}2a)$$

$$\frac{Y}{4} = \frac{AO}{2}\cdot\left(\frac{SUM}{4}\right)\cdot 2 + \frac{A1}{2}\cdot\left(\frac{SUM+1}{4}\right)\cdot 2 + \frac{A2}{2}\cdot\left(\frac{SUM+2}{4}\right)\cdot 2 \;, \qquad (6\text{-}2b)$$

and the data is stored in memory as

$$\begin{aligned} SUM &= w_n \\ SUM+1 &= w_{n-1} \\ SUM+2 &= w_{n-2} \\ Y &= y_n \;, \end{aligned}$$

where w is an intermediate computational variable.

Equations (6-2) are identical in form to the general second-order difference equations for the canonical form, Eqs. (4-7). The first term on the right side of Eq. (6-2a) is labeled SUM only for convenience in the program. This location contains the current input value at the time line 28 is executed. Also since Y/4 is the answer obtained from the program, P036 in Fig. 5-1 must be set to 0.4 to complete the magnitude scaling of the output.

The actual input and output waveforms for this filter are shown in Fig. 6-1a. The ripple in the output wave is caused by the presence of the sampling frequency and its harmonics. These high frequencies could be eliminated by additional analog filtering if a smooth waveform is required. Figure 6-1b shows the attenuation of the output at 200 Hz. The complete frequency response of the filter is exactly identical to the frequency response calculated for the transfer function.

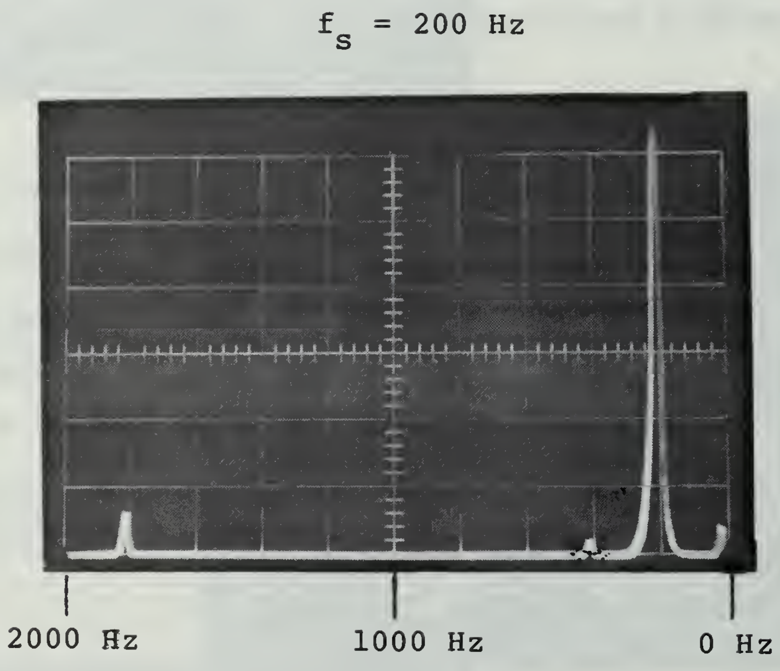


Figure 6-2. The Spectrum of the Output Signal.

That response was plotted in Fig. 3-3. The phase shift caused by the insertion of the filter is also evident in Fig. 6-1.

Figure 6-2 shows the spectrum of the output waveform for an input signal frequency of 200 Hz. The presence of the spectral component at 1800 Hz (2000 Hz - 200 Hz) verifies that the frequency spectrum of the filtered signal is indeed periodic with a period equal to the sampling frequency.

## B. A BANDPASS FILTER

A bandpass digital filter was designed in Section III, C. The filter characteristics included a flat passband from 200 Hz to 500 Hz for a sampling frequency of 2000 Hz. The transfer function of the filter is

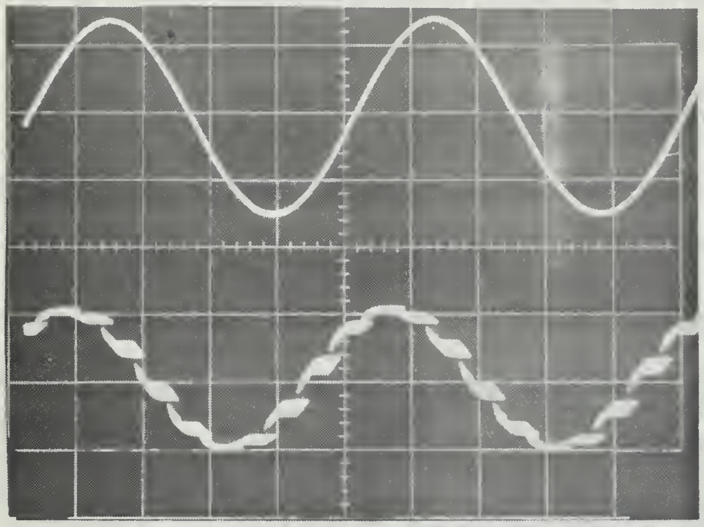
$$H(z) = \frac{0.131106-0.262212z^{-2}+0.131106z^{-4}}{1-1.400064z^{-1}+1.272216z^{-2}-0.658419z^{-3}+0.272217z^{-4}} . \quad (6\text{-}3)$$

### 1. Simulation of the Direct Form

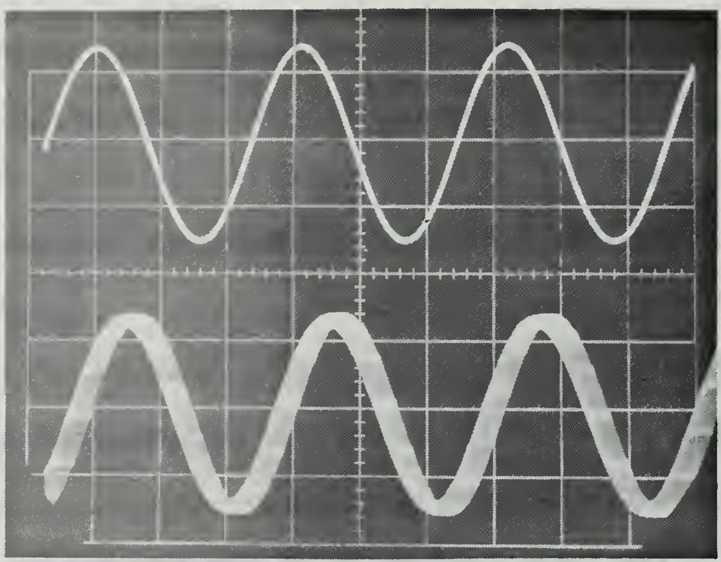
Computer Program 4 gives the abbreviated coding for a direct form of this filter. The difference equation, including the in-program scaling of line 23,

$$\begin{aligned}\frac{SUM}{4} = {} & \frac{A0}{2}\cdot\left(\frac{DATA}{4}\right)\cdot 2 + \frac{A1}{2}\cdot\left(\frac{DATA+1}{4}\right)\cdot 2 + \frac{A2}{2}\cdot\left(\frac{DATA+2}{4}\right)\cdot 2 \\ & + \frac{A3}{2}\cdot\left(\frac{DATA+3}{4}\right)\cdot 2 + \frac{A4}{2}\cdot\left(\frac{DATA+4}{4}\right)\cdot 2 + \frac{B1}{2}\cdot\left(\frac{SUM+1}{4}\right)\cdot 2 \qquad (6\text{-}4) \\ & + \frac{B2}{2}\cdot\left(\frac{SUM+2}{4}\right)\cdot 2 + \frac{B3}{2}\cdot\left(\frac{SUM+3}{4}\right)\cdot 2 + \frac{B4}{2}\cdot\left(\frac{SUM+4}{4}\right)\cdot 2 .\end{aligned}$$

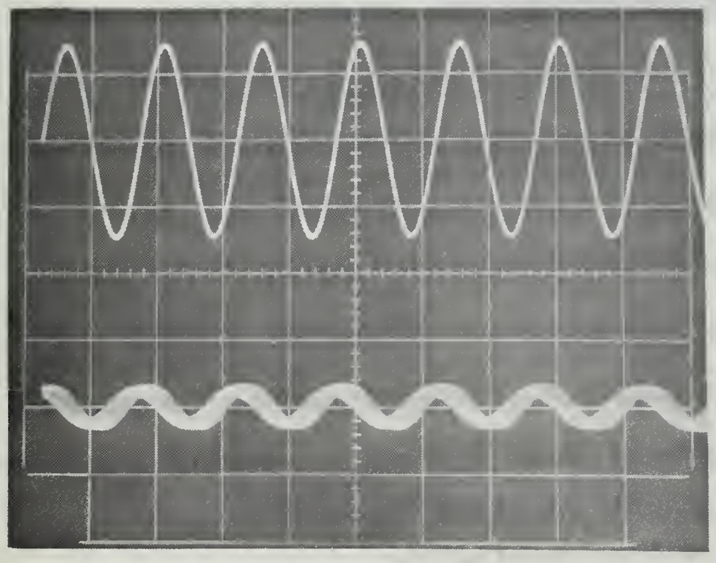
Figure 6-3 shows the output waveforms of this filter at several frequencies. The complete frequency characteristic



$f_s$ = 200 Hz
Time = 1 msec/div
Amplitude = 5V/div



$f_s$ = 320 Hz
Time = 1 msec/div
Amplitude = 5V/div



$f_s$ = 650 Hz
Time = 1 msec/div
Amplitude = 5V/div

Figure 6-3. Several Output Signals from a Bandpass Digital Filter.

does agree with the plot of the transfer-function characteristic given in Fig. 3-4.

2. Simulation of the Cascade Form

The cascade filter form requires the factorization of the digital transfer function into a product of second-order subfilters. The result for the present example is

$$H(z) = \frac{(0.362085-0.724170z^{-1}+0.362085z^{-2})^2}{(1-1.222023z^{-1}+0.603825z^{-2})(1-0.178041z^{-1}+0.450821z^{-2})}. \tag{6-5}$$

The implementation of the cascade form involves the routing of an input signal through a cascade of subfilters (two in this case). If the same basic second-order filter is used for all the subfilters, then when the signal comes out of one subfilter it must be routed around to the input of the common filter as the filter coefficients are changed. This is the most appealing feature of the cascade form. High-order filters can be implemented by time multiplexing a low-order filter section.

The real-time simulation scheme can be programmed to do the same thing. Refer to Computer Program 5. A basic second-order canonical-form filter is coded with an additional indirect addressing feature. Index register 3 adds either one or zero (depending on which of the two subfilter iterations is in progress) to all operands which are followed by the characters ",3". The data is stored in memory as follows:

```
SUM   = output of first subfilter = input to second subfilter
SUM   = input sample = input to first subfilter
SUM+1 = delayed sum of second subfilter
```

SUM+1 = delayed sum of first subfilter
SUM+2 = delayed sum of second subfilter
SUM+2 = delayed sum of first subfilter
Y = final output of complete filter
Y = output of first subfilter,

and the coefficients are stored as shown in the program. The sequence of instructions from "REPEAT" to "BRX REPEAT" are the basic arithmetic instructions for a second-order canonical filter structure. This sequence is executed twice during each sample period. Indirect addressing causes the proper storage locations and filter coefficients to be used for each subfilter. In this way a time-multiplexed filter is simulated. Care must be taken to insure the proper shifting of the data between sample intervals. Refer to lines 51 through 56 in the program.

The bandpass digital filter under consideration was simulated using Computer Program 5. The resultant frequency response was identical to both the calculated response and the response of the direct-form simulation.

3. Simulation of the Parallel Form

The parallel form of a digital filter may be chosen if the paramount consideration is speed in lieu of conservation of hardware. In this scheme the total response is the sum of the responses to the individual subfilters which are arranged in parallel. Refer to Fig. 4-4. Evidently the calculations for each subfilter can be performed simultaneously if no multiplexing of hardware is desired.

The implementation of the parallel-filter structure requires the partial fraction expansion of the digital transfer function into a summation of second-order transfer functions. This operation may involve considerable labor if the order of the filter is high. Computer Program 6 is a relatively simple FORTRAN program for obtaining the partial fraction expansion of some rational polynomials. The multiplicity of poles of the rational function cannot exceed two. Fortunately digital transfer functions normally do not have multiple poles, so the program can be used. Minor modifications to the program would permit expansions of arbitrarily high-order transfer functions.

The bandpass filter presently being considered is of fourth order. Accordingly, the transfer function given in Eq. (6-3) must be expanded into the form

$$H(z) = \gamma + \frac{\alpha_1(z)}{\beta_1(z)} + \frac{\alpha_2(z)}{\beta_2(z)} . \qquad (6\text{-}6)$$

The algebra will not be given here. The results are:

$$\begin{aligned}
\gamma &= 0.131106 \\
\alpha_1(z) &= -0.383754z + 0.149628 \\
\beta_1(z) &= z^2-1.222023z + 0.603825 \\
\alpha_2(z) &= 0.567310z + 0.046308 \\
\beta_2(z) &= z^2-0.178041z + 0.450821.
\end{aligned}$$

Notice that the expansion was made for H(z) equal to a function of z rather than $z^{-1}$. Then, after Eq. (6-6) is written in standard form (a function of $z^{-1}$), the filter and program coefficients become

| | | | |
|---|---|---|---|
| $\gamma$ = | 0.131106 | GAMMA = | 549898 |
| $a_{11}$ = | -0.383754 | $A1_1$ = | -1609581 |
| $a_{12}$ = | 0.567310 | $_2$ = | 2379471 |
| $a_{21}$ = | 0.149628 | $A2_1$ = | 627585 |
| $a_{22}$ = | 0.046308 | $_2$ = | 194230 |
| $b_{11}$ = | -1.222023 | $B1_1$ = | 5125536 |
| $b_{12}$ = | -0.178041 | $_2$ = | 746758 |
| $b_{21}$ = | 0.603825 | $B2_1$ = | -2532626 |
| $b_{22}$ = | 0.450821 | $_2$ = | -1890880 |

Computer Program 7 is the simulation of the parallel form of the bandpass digital filter. The program is identical to the program for the cascade form except that the outputs of each individual subfilter are stored separately and summed together at the end. The simultaneous operation of the subfilters obviously cannot be simulated because the computer's single arithmetic unit must be time-multiplexed. The correct results for the parallel filter are obtained although the increase in operating speed is not. In fact, the parallel filter simulation takes slightly longer than the cascade simulation because of this additional summation sequence (lines 49 through 53) which is required at the end.

The results of the real-time simulation of the bandpass filter using Computer Program 7 were essentially identical to those of the previous two simulations.

## 4. Experimental Observations of Quantization Effects

The multiplying coefficients were calculated to six-decimal-place accuracy in the design of the bandpass digital filter. This accuracy is retained by the 23-bit magnitude representation in the XDS 9300 computer. If the least significant bits of the coefficient data words are systematically truncated, some insight into the effects of coefficient quantization can be obtained. A software package is available with the XDS 9300 computer which permits altering the contents of memory locations via manual input to the teletypewriter.

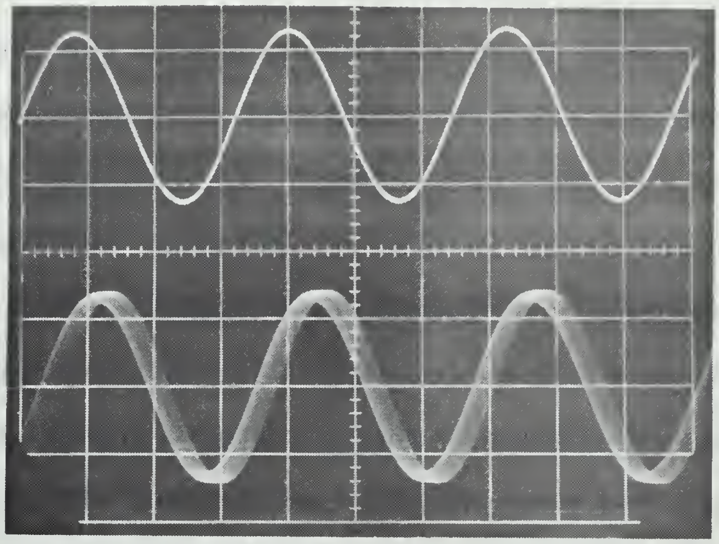
The coefficients for each of these bandpass-filter configurations were truncated in steps of single octal integers with the proper rounding being performed. Generally the effect of the less accurate coefficients was to gradually alter the frequency-response characteristic until it no longer satisfied the filter specifications. This occurred when the program coefficients were reduced to three significant octal digits in each case. Thus, eight magnitude bits or third-decimal-place accuracy was required for proper operation of the bandpass digital filter.

The results of this example do not permit any general conclusions to be made regarding the choice of one filter form over another. It is likely that the arguments against the direct form from the point of view of coefficient accuracy which are made in the literature [1, 4, and 5] have greater significance in higher-order filters. Also the problem of filter stability was not encountered in this example, since the poles of the digital transfer function were located well within the unit circle in the z-plane.

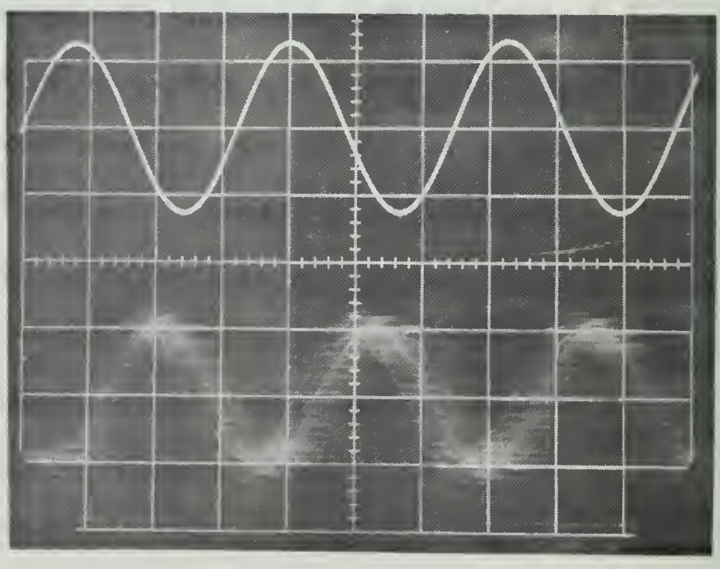
Modification of the computer word length is accomplished in the simulation program by inserting the ETR instruction after each multiplication and altering the contents of location MASK as desired. Although the A/D converter provides only 13 bits of accuracy in the input data, all calculations are performed with 24-bit accuracy unless altered by the ETR instruction. Accordingly the effects of round-off error in a digital filter can be observed as the computer word length is reduced.

The contents of location MASK were decremented from 15 bits for each filter form. The point at which quantization noise reduces the response of the digital filter to an unsatisfactory state must be judged in light of the filter application. For this example eight bits and seven bits were required for the direct form and the parallel form respectively to retain basic signal definition on an oscilloscope. The response of the cascade filter was unsatisfactory even when the computer word was truncated to 15 bits. Figure 6-4 shows the output waveform for a 300 Hz signal using 15-bit arithmetic in the cascade and parallel filters. Figure 6-5 shows the relative quantization noise in the spectra of the two signals.

It appears that the cascade filter structure, with its serial form of signal processing, requires considerably greater accuracy in arithmetic computations than either of the other two structures. Hence, the significant advantage of using time-multiplexed circuitry is somewhat offset by the increased word length requirement.



$f_s$ = 300 Hz
Parallel Form



$f_s$ = 300 Hz
Cascade Form

Figure 6-4. Quantization Effects in Digital Filters

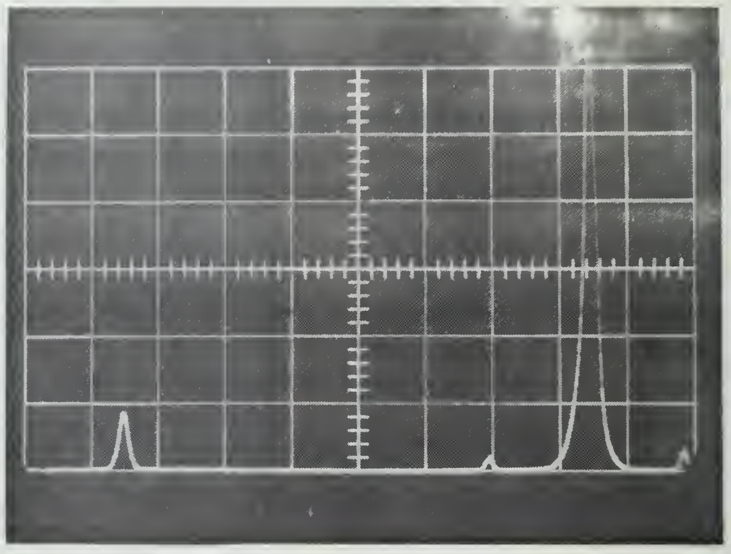
## C. A NOISE FILTER

As a final example consider the problem of generating a noise signal whose power spectrum is of uniform height within a specified frequency range, but is significantly attenuated outside this frequency range. Such a noise signal has application in various signal-analysis and measurement schemes for communications systems, radar systems, etc.

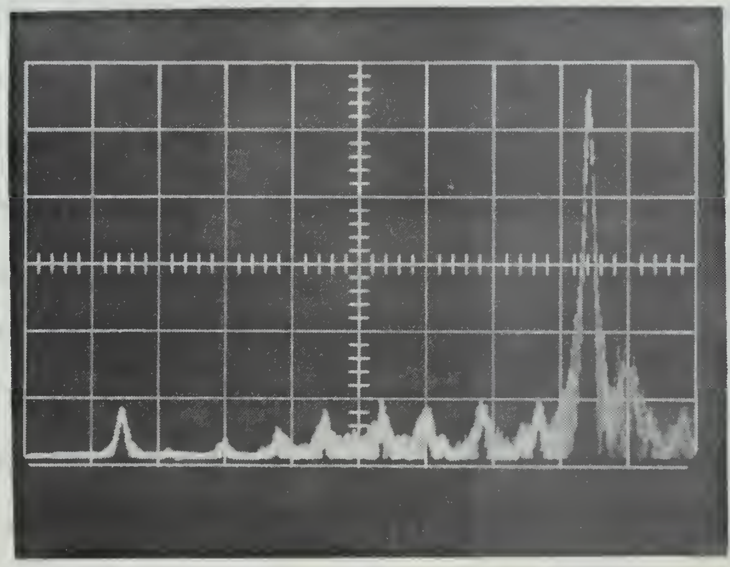
If a continuous-wave carrier signal is frequency-modulated by a gaussian noise source, the resultant signal spectrum is gaussian in shape and centered on the carrier frequency. The desired noise spectrum could be obtained if this gaussian-shaped spectrum could be filtered by a bandpass filter whose magnitude-versus-frequency characteristic varied as the reciprocal of a gaussian curve within the passband.

Such a problem in spectral shaping can be solved using a digital filter. The bandpass digital filter which was presented in Section III, C, as an example of matched z-transform design is offered as an approximate solution to the problem.

The gaussian spectrum was generated by applying a low-frequency noise source (the amplitude distribution of the noise voltage is gaussian) to a voltage-controlled oscillator. The carrier frequency was chosen to be 450 Hz and the rms voltage of the noise was 1.0 volts. Applying a constant 1.0 volts to the VCO displaced the carrier frequency 200 Hz. Therefore the gaussian spectrum centered at 450 Hz has a standard deviation of 200 Hz.



$f_s$ = 300 Hz
Parallel Form



$f_s$ = 300 Hz
Cascade Form

Figure 6-5. Quantization Effects in Signal Spectrums.

The bandpass second-order Chebyshev filter was designed for a 12-db gain between the peaks and the valley. This corresponded to a displacement of 1.6 standard deviations from the mean on the normal curve; or, in terms of frequency, for the given hardware it corresponds to the peak frequencies being separated from the carrier by 320 Hz. Thus, the ideal frequency characteristic for the specified conditions is shown in Fig. 6-6. The actual solution obtained by standard

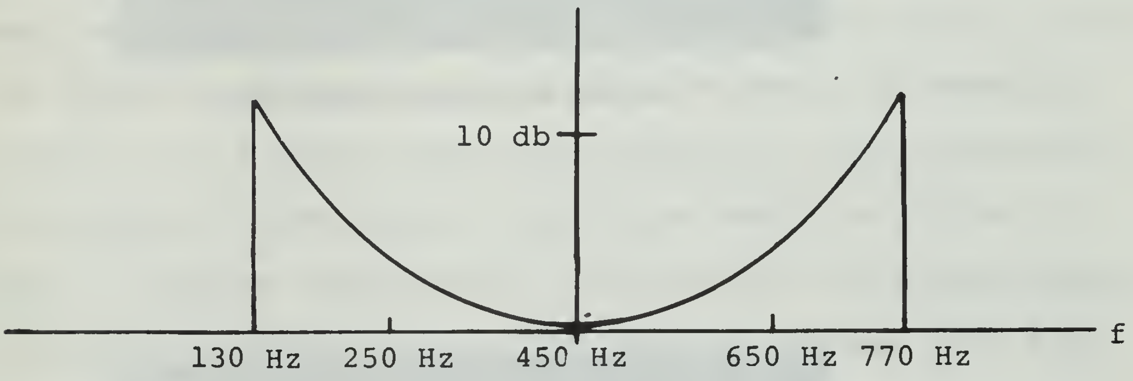
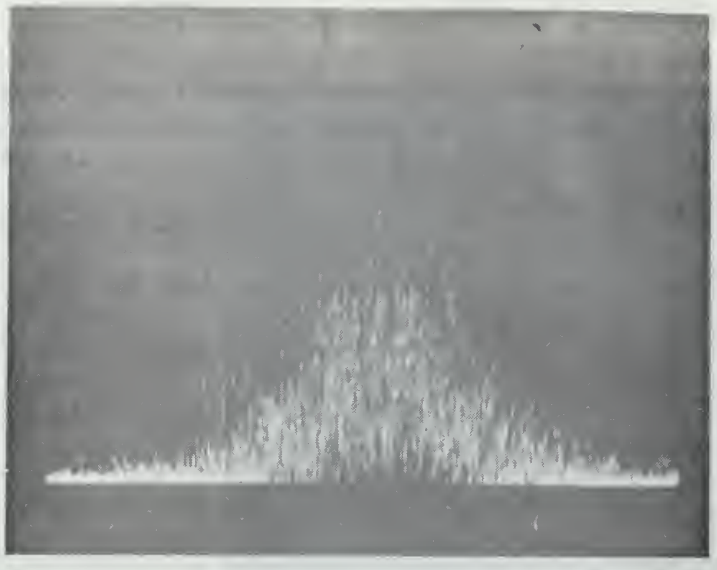


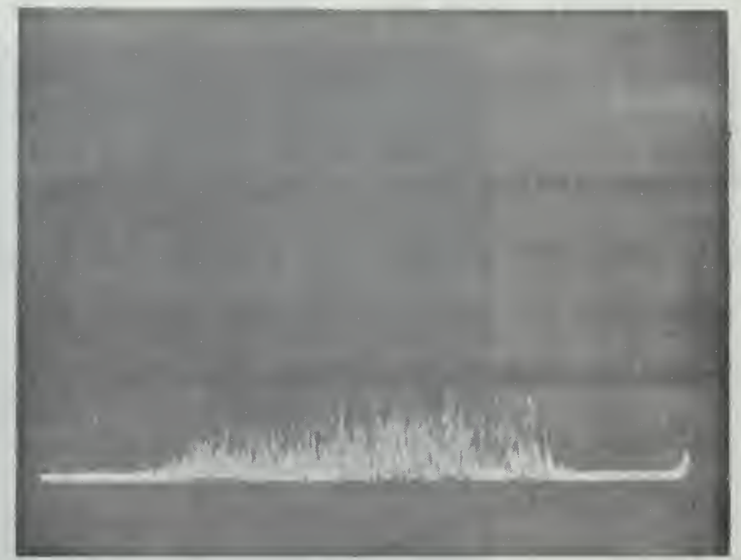
Figure 6-6. An Ideal Noise Filter.

analog-filter design methods and a subsequent matched z-transformation was plotted in Fig. 3-7. Notice that the s-domain solution in Fig. 3-6, which is itself distorted by the low-pass-to-bandpass transformation, suffers further distortion by the matched z-transformation. The final result is, however, a fair approximation of the desired frequency characteristic.

The gaussian spectrum, centered at 450 Hz, is shown in Fig. 6-7a and the filtered spectrum is shown in Fig. 6-7b. Notice that the gain factor in the digital filter was adjusted for 6db gain at the peak frequencies and a 6-db attenuation



a. Gaussian Input Spectrum



b. Filtered Output Spectrum

Figure 6-7. Output Signal and Spectrum from a Digital Noise Filter,

at the center frequency. Further refinement in this filter design could be accomplished using optimization techniques in the s domain.

# VII. SUMMARY AND CONCLUSIONS

The subject of digital signal processing continues to receive ever increasing attention. The basic appeal of digital circuitry is still lower cost, higher reliability and faster operation. Specifically, real-time digital filters now appear to be on the threshold of becoming a commercial reality. To a large extent the impetus for the theoretical development of the digital filter has been the promise of special-purpose computational hardware using LSI technology. It appears extremely likely that digital filters will find significant real-time applications in the not too distant future.

An attempt has been made in this thesis to assemble and organize much of the current theory of digital filters, and to present in detail those aspects relating to the real-time implementation of digital filters. Furthermore, the proposed hybrid-computer simulation scheme provides a vehicle to process signals in real time and to observe the theory of digital filters in action.

## A. COMMENTS ON DESIGN METHOD AND IMPLEMENTATION TECHNIQUES

Most of the theory of the z-transform and its application to discrete-time signals is not new. However, the development of the digital filter has brought about new variations of that transform. For example, discussions of the bilinear and the matched z-transforms are not found in the older texts on sampled-data systems. Also the very popular digital-filter design technique of using s-plane continuous-filter synthesis has generated new and extremely useful "special purpose" transformations. One of these transformations, the bandpass version of the bilinear z-transform, obviates the conventional low-pass-to-bandpass frequency transformations in the s-plane.

The usefulness of all the various transformations lies in their ability to be easily implemented and to produce accurate results. The use of large computer programs to synthesize digital transfer functions from continuous transfer functions adds versatility to the indirect design methods.

The three general implementation schemes which have been discussed in this paper all have useful applications. The direct form (including both the straight and canonical versions) is the simplest to use and gives very satisfactory results for relatively low-order filters. The cascade form represents a high-order filter as a cascade of low-order subfilters. This scheme is particularly adapted to realization in special-purpose hardware because of the "basic filter section" concept. A second-order filter section is time-multiplexed to permit the cascading of a single signal through the filter section n times. In this way a 2nth-order filter can be realized.

Time multiplexing is the key to the economical use of hardware. For example, the serial multiplier which is presented in Appendix A would seem to be a disproportionately complex circuit were it not for the built-in ability to handle multiplexed input signals and coefficients.

The final filter structure is the parallel form. This form could provide the fastest signal processing if separate hardware were available for each subfilter. On the other hand, the parallel structure can also make use of time-multiplexed hardware. This implementation scheme retains the compact and economical construction of the cascade form, but could operate with a shorter word length. This is because the cascade form inherently requires greater accuracy in the arithmetic unit.

## B. AN APPRAISAL OF THE REAL-TIME DIGITAL-FILTER SIMULATION

The hybrid-computer program which was presented in this paper represents a first step toward the real-time processing of signals by digital filters. Theoretically any digital filter that is designed for real-time application could be simulated by this general scheme. (Real-time digital filters are considered to be of the recursive type in this context.)

In the author's view there are two basic limitations to the digital-filter simulation scheme. The first is the 24-bit word length of the XDS 9300 computer. The coefficient and arithmetic accuracy required in high-order filters may not be attainable using a 24-bit computer word. A solution to this problem would be to rewrite the simulation programs for double-precision arithmetic.

The second limitation is that there is an upper limit on the allowable sampling frequency imposed by the cycle time of the XDS 9300 computer. Execution of a second-order difference equation (one basic filter section) in Computer Program 7 requires approximately 200 microseconds of computer time. This figure varies slightly depending on the arithmetic sequence being used. This minimum execution time means that a fourth-order filter cannot be simulated using a sample frequency higher than 2500 Hz, or that an eighth-order filter cannot be simulated using a sample frequency higher than 1250 Hz, etc. From these figures it is evident that experimentation must be limited to the sub-audio range of frequencies. Indeed, the same general problem applies to all digital-filter systems.

A partial solution to the problem at the Naval Postgraduate School is to write a more efficient program for the XDS 9300 computer. However, the future for the general simulation scheme and for digital filters as a whole is much brighter. The next generation of digital computers will make extensive use of LSI technology. Computers using monolithic circuits will have addition times in the order of 50 nanoseconds vice 1.75 microseconds for the XDS 9300 computer. This reduction of almost two orders of magnitude in computing speed will mean an expansion of the digital-filtering spectrum to include the complete audio range and beyond. There is every reason to believe that special-purpose hardware will benefit from similar advances.

# APPENDIX A

# A BINARY SERIAL MULTIPLIER FOR APPLICATIONS IN DIGITAL FILTERS

## 1. INTRODUCTION

Real-time implementation of digital filters involves the construction of a special-purpose digital computer. The arithmetic unit of this computer, which performs the multiplication and addition of digital signals, presents some interesting challenges in logical design. One part of the arithmetic unit, the multiplier, is the subject of this Appendix.

Specifically the computational aspects of an arithmetic unit which are peculiar to digital filters are discussed. From these considerations some general specifications for a multiplier are determined. Finally one solution to the problem is presented using a binary serial multiplier.

## 2. COMPUTATIONAL CONSIDERATIONS IN DIGITAL FILTERS

Most digital-filter transfer functions permit simultaneous arithmetic operations during a given Nyquist interval. If the number of operations to be performed is large, economy can be achieved using serial arithmetic and by sharing the adders and multipliers. Time sharing in the filter can be accomplished with multiplexing circuitry. In addition to a significant simplication of the hardware, serial arithmetic provides increased modularity and flexibility in the digital circuit configurations. Also the processing rate is limited only by circuit speed and not by carry-propagation times in the adders and multipliers. Finally with serial arithmetic sample delays are realized simply as single-input single-output shift registers.

The two's complement representation of binary numbers is also appropriate because addition can proceed starting with the least significant bit with no advance knowledge of the signs and with no later correction to sums being necessary. For example, the algebraic summation of the two signed numbers, 1.101 and 0.010, can be accomplished as follows:

$$\begin{array}{lcl} & \begin{array}{c}\text{conversion to}\\ \text{two's complement}\end{array} & \\ \text{Subtract}\quad \begin{array}{r}1.101\\ \underline{0.010}\\ 0.011\end{array} & \longleftrightarrow & \begin{array}{r}0.011\\ \underline{1.010}\\ 1.101\end{array}\quad \text{Add}\end{array}$$

Addition using the two's complement form gives the correct signed result with no concern about signs or carries. In the multiplier, however, it is simpler to treat the sign separately and multiply only positive numbers. After the positive product is obtained the correct sign is reinserted in the proper bit location.

Multiplication in digital filters is subject to a very severe restriction. Normally the multiplication of an N-bit number by a K-bit number results in a product (N+K) bits long. This situation cannot be tolerated in a serial system because the N-bit words (representing successive samples) are adjacent to each other in time. Consequently there is no room for the generation of a product in excess of N bits. If an N-bit sample is multiplied by a K-bit coefficient, the product must be truncated and rounded to N bits. The solution to this problem is to multiply one bit at a time using partial sums. It can be shown that the truncation of each partial sum and rounding the final partial sum will produce the same result as truncating and rounding a full final product.

<u>EXAMPLE</u>

Let x = the sample value = 0.00101, where the last bit in time is the sign bit (0 always).
a = the multiplying coefficient = 1.0110 with no sign bit.

In this example N=6 and K=5. Then multiplication in the normal fashion can be compared to partial sum multiplication as follows:

```
                 NORMAL                        PARTIAL SUM
              ___________                     ___________
              0.0 0 1 0 1                     0.0 0 1 0 1
                1.0 1 1 0                       1.0 1 1 0
              ___________                     ___________
              0 0 0 0 0 0                     0 0 0 0 0 Ø
            0 0 0 1 0 1                     0 0 0 1 0 1
          0 0 0 1 0 1                       ___________
        0 0 0 0 0 0                         0 0 0 1 0 1̸
      0 0 0 1 0 1                         0 0 0 1 0 1
      _____________________               _____________
      0 0 0 1 1 0 1̸ 1̸ 1̸ Ø                 0 0 0 1 1 1̸
                1                       0 0 0 0 0 0
      _____________________             _______________
      0.0 0 1 1 1                       0 0 0 0 1 1̸
                                      0 0 0 1 0 1
                                      _________________
                                      0 0 0 1 1 0
                                                1
                                      _________________
                                      0.0 0 1 1 1
```

The results are identically correct.

If we require the magnitude of the sample to be less than one (i.e., $|x|<1.0$), and the magnitude of the multiplying coefficient to be less than two (i.e., $|a|<2.0$), then the fixed-point multiplication of the positive coefficient times the positive sample will always be of the form shown in the example. That is, the product is always positive and the decimal point follows the sign bit just as it did in the input to the multiplier. It is important to note that the sign bit of the coefficient is not included in the multiplication. The restrictions on magnitude size are not unrealistic because the sample can easily be scaled. Also the multiplying coefficients for typical cascade and parallel filter forms are in the range -2.0 to 2.0. Any overflow in the partial sums will cause an error. This problem can be eliminated by further scaling and will not be discussed.

In view of the above discussion the general requirements for the multiplier are the following:

1. Accept an uncomplemented sample value from a serial delay device. This sample will be in binary form, N bits long, with the last bit in time representing the sign. The remainder of the word will be a fractional value less than one.

2. Accept each single bit of a K-bit multiplying coefficient separately for use as a constant bit multiplier. The sign bit is used only to determine the sign of the product and the magnitude of the coefficient is less than 2.0.

3. Remove the sign bit, $x_N$, from the sample value and multiply it by the coefficient sign bit, $a_{K+1}$, modulo 2 (exclusive-OR) to determine the sign of the product.

4. Multiply the remaining positive sample by the positive multiplying coefficient using a serial multiplication scheme. The partial products must be truncated to N bits during each step and the final N-bit product is rounded up by adding in the last truncated bit. The sign bit of this product must always be a zero (positive).

5. Reinsert the proper sign bit into the final product. The decimal point will be positioned immediately after the sign bit in the product.

6. Convert the signed product to the two's complement form for insertion into the following adder.

## 3. THE MULTIPLIER

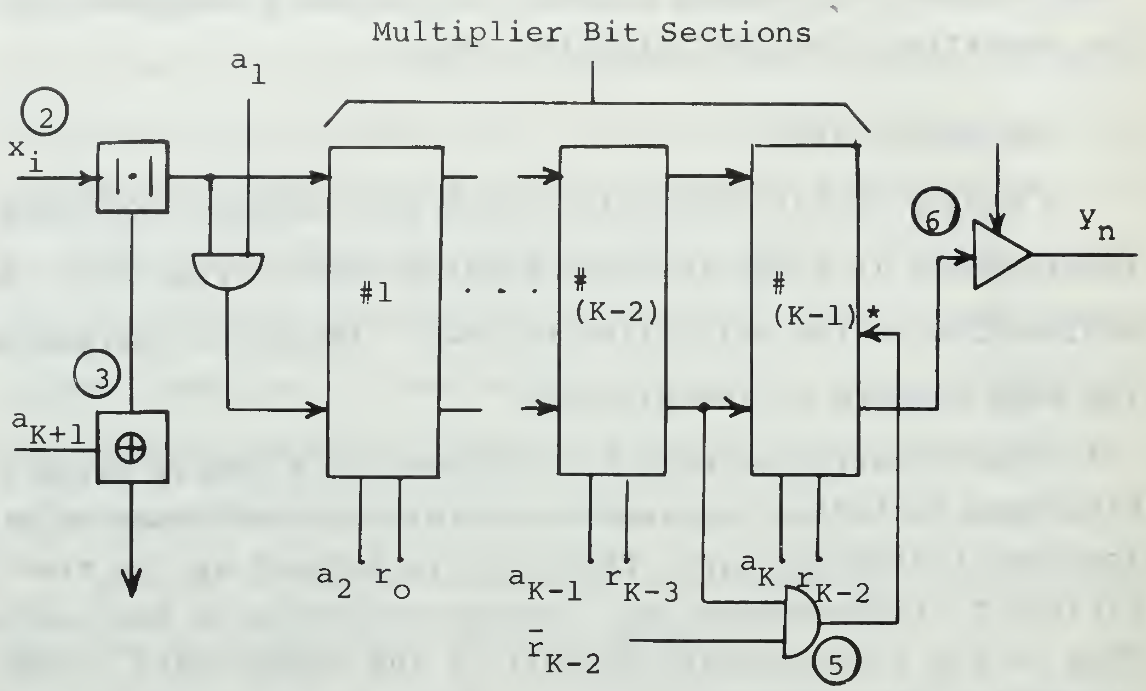
A binary serial multiplier which will satisfy the above requirements is shown in block diagram form in Fig. A-1. An explanation of the multiplier follows: (Refer to corresponding step numbers on the diagram).

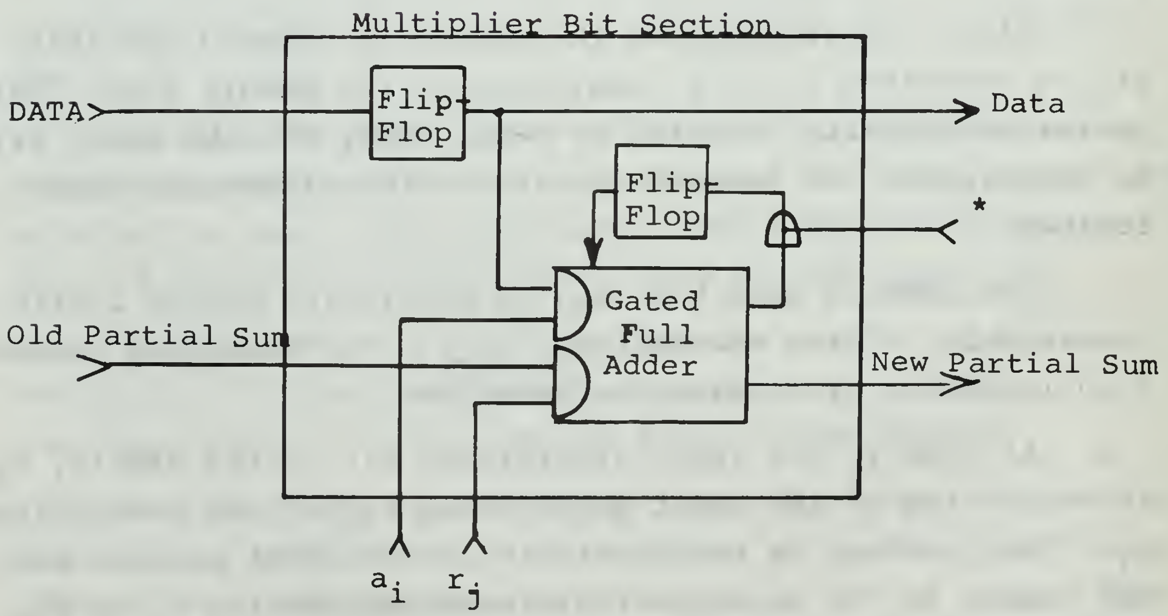
1. The notation of Fig. A-1 is based on a sample value N bits long including sign and a multiplying coefficient K bits long not including sign. Time "$t_o$" is defined as the time the first bit of the sample, $x_1$, becomes available at the input. This is the least significant bit of the sample word. Then succeeding times become $t_1$, $t_2$, $\cdots t_{N-1}$ (which is the time the last bit of the sample, $x_N$, becomes available at the same point).

2. First the sign bit of the sample is removed and that bit is converted to zero (positive) in the sample word. This operation normally requires an N-bit delay but the delay will be disregarded for sake of clarity in the discussion which follows.

3. The removed sign bit, $x_N$, is multiplied modulo 2 with the sign bit of the coefficient, $a_{K+1}$. The resulting product sign is stored or delayed for later use.

4. At time $t_o$ the least significant bit of the sample, $x_1$, is multiplied by the least significant bit of the coefficient, $a_1$. This product is the first bit of the first partial sum. The inputs to the multiplier bit sections labeled $r_j$ (j=0,1, $\cdots$K-2) are timing signals which accomplish the truncation. At time $t_o$, $r_o$ would be low (the other $r_j$ being high) so the output of partial sum 1 at time $t_o$ would be gated to zero. At time $t_1$ $r_1$ goes low so the output of partial sum 2 at $t_1$ is zero. In this fashion the signal at the input to the





Note: $r_j$ is low at bit time j. The line is high all other times.

Figure A-1. A Binary Serial Multiplier.

multiplier follows the sequence $x_1(t_o)$, $x_2(t_1)$, $\cdots x_N(t_{N-1})$ while the signal at the output follows the sequence $0(t_o)$, $0(t_1)$, $\cdots 0(t_{K-2})$, $P_1(t_{K-1})$, $P_2(t_K)$, $\cdots P_N(t_{K+N-2})$. To illustrate the scheme more clearly consider the following example:

<u>EXAMPLE</u>

Let $x=0.010$ $\qquad N=4$

$\quad a=1.011$ $\qquad K=4$

$$\begin{array}{llccccccccl}
\text{Then} & x = & & & & 0. & 0 & 1 & 0 & \\
 & a = & & & & 1. & 0 & 1 & 1 & \\
\cline{6-9}
 & & & & & 0 & 0 & 1 & 0 & \text{1st partial sum} \\
 & & & & 0 & 0 & 1 & 0 & & \\
\cline{5-9}
 & & & & 0 & 0 & 1 & \cancel{1} & & \text{2nd partial sum} \\
 & & & 0 & 0 & 0 & 0 & & & \\
\cline{4-9}
 & & & 0 & 0 & 0 & \cancel{1} & & & \text{3rd partial sum} \\
 & & 0 & 0 & 1 & 0 & & & & \\
 & & & & & 1 & & & & \text{Last truncated bit} \\
\cline{3-6}
 & y = & 0. & 0 & 1 & 1 & & & & \text{4th partial sum} \\
 & & t_6 & t_5 & t_4 & t_3 & t_2 & t_1 & t_0 & \\
\end{array}$$

$$t_6 t_5 t_4 t_3 \underbrace{t_2 t_1 t_0}_{(K-1)\text{ bit delay in product.}}$$

It is easily seen that the preceding sample, occurring at the input times $t_{-4}$, $t_{-3}$, $t_{-2}$, $t_{-1}$ would have produced a correct product at the output at times $t_{-1}$, $t_o$, $t_1$, $t_2$.

5. In the last multiplier bit section, the bit which is truncated at bit time $t_{K-2}$ is fed back through the carry loop to accomplish the rounded product.

6. Finally the sign bit (zero in all cases except overflow) is changed to the correct value and the correctly signed product is then converted to two's complement form for insertion into an adder.

It is seen from Fig. A-1 and the above discussion that this serial multiplier has an inherent delay of (K-1) bits, not counting any delay encountered in the sign bit removal. This additional delay must be deducted from the prescribed delay in the multiplication branch for proper operation of the digital filter.

COMPUTER PROGRAM 1

```
C     FREQUENCY RESPONSE OF A DIGITAL TRANSFER FUNCTION
C
C     M IS THE NUMBER OF CASCADED SECOND-ORDER TRANSFER
C     FUNCTIONS
C     F1 AND F2 ARE THE LOW AND HIGH FREQUENCIES IN THE
C     FREQUENCY RANGE OF INTEREST
C     T IS THE SAMPLING INTERVAL
C     MAGH IS THE MAGNITUDE OF THE TRANSFER FUNCTION
C     EVALUATED AT Z=EXP(-J*OMEGA*T)
C     MAGNH IS MAGH EXPRESSED IN DECIBELS
C
      DIMENSION A00(10),A11(10),A22(10),B11(10),B22(10)
      REAL MAGH,MAGNH,NUM1,NUM2
      INTEGER F1,F2
      COMPLEX NUM,DENOM
      READ(5,200)M,F1,F2,T
  200 FORMAT(I2,2I4,F10.0)
      WRITE(6,100)
  100 FORMAT(1H1,10X,'FREQUENCY',5X,'ABS GAIN',6X,'GAIN(DB)'
     */)
      DO 5 I=1,M
      READ(5,101)A00(I),A11(I),A22(I),B11(I),B22(I)
  101 FORMAT(5F10.0)
    5 CONTINUE
      F=F1
      DO 10 I=F1,F2
      X=F*6.28318530 6*T
      MAGH=1.0
      DO 6 J=1,M
      A0=A00(J)
      A1=A11(J)
      A2=A22(J)
      B1=B11(J)
      B2=B22(J)
      NUM1=(A2+A0)*COS(X)+A1
      NUM2=(A0-A2)*SIN(X)
      NUM=CMPLX(NUM1,NUM2)
      DENOM1=(B2+1.)*COS(X)+B1
      DENOM2=(1.-B2)*SIN(X)
      DENOM=CMPLX(DENOM1,DENOM2)
      MAGH=ABS(MAGH*CABS(NUM/DENOM))
    6 CONTINUE
      MAGNH=20.*ALOG10(MAGH)
      WRITE(6,102)F,MAGH,MAGNH
  102 FORMAT(1H ,09X,F6.1,8X,F8.6,7X,F8.3)
      F=F+1.0
   10 CONTINUE
      STOP
      END
```

# COMPUTER PROGRAM 2

```
                                 *  DIGITAL FILTER SIMULATIØN
                                 *  SECØND-ØRDER LØW-PASS FILTER (DIRECT FØRM)
                                 *
         00000005                  A       EQU    5
00000    0 2 17 00143              START   LDX    =077700005,2    SET STØRAGE LØCATIØNS
00001    0 40 0 377 01                     CØPY   (0,5)           TØ ZERØ
00002    0 2 76 00131                      STA    DATA,2
00003    0 2 57 00002                      BRX    $-1,2
00004    0 0 16 00121                      LDA    BRMAD           INITIALIZE HARDWARE FØR A/D
00005    0 0 76 00040                      STA    040
00006    0 0 16 00122                      LDA    BRMDA           INITIALIZE HARDWARE FØR D/A
00007    0 0 76 00041                      STA    041
00010    0 0 16 00123                      LDA    BRMHLT          INITIALIZE HARDWARE FØR
00011    0 0 76 00055                      STA    055             CØMPLETIØN ØPS
00012    0 0 20 31020                      SKS    031020          START TEST ØCCURS ØN T-104
00013    0 0 01 00012                      BRU    $-1
00014    00220002                          EIR                    ENABLE INTERRUPTS
00015    0 02 0 33004                      EØM    033004
00016    0 0 20 31010              GØ      SKS    031010          END TEST ØCCURS ØN T-103
00017    0 0 01 00111                      BRU    ENDRUN
00020    0 0 03 00057                      BRM    ADØPS           TAKE SAMPLE
00021    0 0 16 00131                      LDA    DATA            DATA IS THE INPUT
00022    0 0 63 00135                      MUL    A0              AT SAMPLE TIME N
00023    0 0 6004 001                      ALSD   1
00024    0 0 76 00126                      STA    SUM             SUM IS ACCUMULATED ØUTPUT
00025    0 0 16 00132                      LDA    DATA+1          AS TERMS ØF DIFFERENCE EQN
00026    0 0 63 00136                      MUL    A1              ARE ADDED TØGETHER
00027    0 0 6004 001                      ALSD   1
00030    0 0 35 00126                      ADM    SUM
00031    0 0 16 00133                      LDA    DATA+2
```

COMPUTER PROGRAM 2 (Continued)

```
00032  0 0 63 00137                MUL    A2
00033  0 0 6004 001                ALSD   1
00034  0 0 35 00126                ADM    SUM
00035  0 0 16 00127                LDA    SUM+1
00036  0 0 63 00140                MUL    B1
00037  0 0 6004 001                ALSD   1
00040  0 0 35 00126                ADM    SUM
00041  0 0 16 00130                LDA    SUM+2
00042  0 0 63 00141                MUL    B2
00043  0 0 6004 001                ALSD   1
00044  0 0 35 00126                ADM    SUM            FINAL SUM FØR SAMPLE TIME N
00045  0 0 03 00067                BRM    DAØPS          PERFØRM D/A ØF ØUTPUT
00046  0 2 17 00144                LDX    =077700004,2   SHIFT DATA TØ PREPARE
00047  0 2 16 00131                LDA    DATA,2         FØR SAMPLE TIME N+1
00050  0 2 76 00132                STA    DATA+1,2
00051  0 2 57 00047                BRX    $-2,2
00052  0 40 0 377 01               CØPY   (0,5)
00053  0 0 76 00125                STA    HLTFLG
00054  0 0 53 00125                SKN    HLTFLG         WAIT FØR SAMPLE TIME N+1
00055  0 0 01 00054                BRU    $-1
00056  0 0 01 00016                BRU    GØ             CØMMENCE ØPS FØR TIME N+1
                            *
00057  0 0 00 00000         ADØPS  PZE                   PERFØRM A/D
00060  0 40 0 377 01               CØPY   (0,5)
00061  0 0 76 00124                STA    ADFLG
00062  0 02 0 34000                EØM    034000
00063  0 0 31 00113                PØT    ADCØN
00064  0 0 53 00124                SKN    ADFLG
00065  0 0 01 00064                BRU    $-1
00066  0 0 41 00057                BRR    ADØPS
                            *
00067  0 0 00 00000         DAØPS  PZE                   PERFØRM D/A
00070  0 0 16 00126                LDA    SUM
```

COMPUTER PROGRAM 2 (Continued)

```
00071  0 0 11 00142            ETR     MASK
00072  0 0 05 00134            ADD     DAADR
00073  0 0 76 00117            STA     DACØM
00074  0 02 0 35000            EØM     035000
00075  0 0 31 00116            PØT     DACØN
00076  0 0 41 00067            BRR     DAØPS
                      *
00077  0 0 00 00000   ADEND    PZE
00100  0 0 73 00124            SKR     ADFLG           WAIT FØR A/D TØ CØMPLETE
00101  0 0 01 00100            BRU     $-1
00102  1 0 57 00077            BRC     *ADEND
                      *
00103  0 0 00 00000   DAEND    PZE                     DØ NØT WAIT FØR D/A
00104  1 0 57 00103            BRC     *DAEND          TØ CØMPLETE
                      *
00105  0 0 00 00000   HLTEND   PZE                     NEW CLØCK SIGNAL STARTS
00106  0 0 73 00125            SKR     HLTFLG          ØPS FØR NEXT SAMPLE PERIØD
00107  0 0 01 00106            BRU     $-1
00110  1 0 57 00105            BRC     *HLTEND
                      *
00111  00220004       ENDRUN   DIR                     DISABLE INTERRUPTS
00112  0 0 01 00013            BRU     GØ-3
                      *
                      CØN      FØRM    9,15
00113  001 00114      ADCØN    CØN     1,ADCØM
00114  004 00131      ADCØM    CØN     4,DATA
00115  0 0 00 00000            PZE
00116  001 00117      DACØN    CØN     1,DACØM
00117                 DACØM    RES     1
00120  0 0 00 00000            PZE
00121  0 0 03 00077   BRMAD    BRM     ADEND
00122  0 0 03 00103   BRMDA    BRM     DAEND
00123  0 0 03 00105   BRMHLT   BRM     HLTEND
```

COMPUTER PROGRAM 2 (Continued)

```
00124                     97 ADFLG   RES     1
00125                     98 HLTFLG  RES     1
00126                     99 SUM     RES     3               ALLØCATE STØRAGE FØR SUM
00131                    100 DATA    RES     3               ALLØCATE STØRAGE FØR DATA
00134  00000005          101 DAADR   DATA    5               D/A ADDRESS
00135  01050457          102 A0      DATA    282927
00136  02121136          103 A1      DATA    565854
00137  01050457          104 A2      DATA    282927
00140  22223226          105 B1      DATA    4794006
00141  71312253          106 B2      DATA    •1731413
00142  77777000          107 MASK    DATA    077777000
       00000000          108         END     START
00143  77700005
00144  77700004
```

# COMPUTER PROGRAM 3

```
*   DIGITAL FILTER SIMULATION
*   SECOND-ORDER LOW-PASS FILTER (CANONICAL FORM)
*
*
*
*
*
*
            BRM      ADOPS           TAKE SAMPLE
            LDA      SUM
            ARSA     2               DIVIDE SAMPLE BY 4
            STA      SUM
            LDA      SUM+1
            MUL      B1
            ALSD     1
            ADM      SUM
            LDA      SUM+2
            MUL      B2
            ALSD     1
            ADM      SUM
            LDA      SUM
            MUL      A0
            ALSD     1
            STA      Y
            LDA      SUM+1
            MUL      A1
            ALSD     1
            ADM      Y
            LDA      SUM+2
            MUL      A2
            ALSD     1
            ADM      Y               OUTPUT VALUE AT TIME N
            BRM      DAOPS           PERFORM D/A OF OUTPUT
            LDX      =077700002,2    SHIFT DATA TO PREPARE
            LDA      SUM,2           FOR SAMPLE TIME M+1
            STA      SUM+1,2
            BRX      $-2,2
*
*
*
*
*
*
*
*
*
*
SUM         RES      3
Y           RES      1
DAACR       DATA     5               D/A ADDRESS
A0          DATA     282927
A1          DATA     565854
A2          DATA     282927
B1          DATA     4794006
B2          DATA     -1731413
MASK        DATA     077777000
            END      START
```

COMPUTER PROGRAM 4

```
*   DIGITAL FILTER SIMULATION
*   FOURTH-ORDER BANDPASS FILTER (DIRECT FORM)
*
*
*
*
            BRM     ADOPS             TAKE SAMPLE
            LDA     DATA
            ARSA    2                 DIVIDE SAMPLE BY 4
            STA     DATA              DATA IS NOW THE INPUT
            MUL     A0
            ALDS    1
            STA     SUM               SUM IS ACCUMULATED OUTPUT
            LDA     DATA+1            AS TERNS OF DIFFERENCE
            MUL     A1                EQN ARE ADDED TOGETHER
            ALSD    1
            ADM     SUM
            LDA     DATA+2
            MUL     A2
            ALSD    1
            ADM     SUM
            LDA     DATA+3
            MUL     A3
            ALSD    1
            ADM     SUM
            LDA     DATA+4
            MUL     A4
            ALSD    1
            ADM     SUM
            LDA     SUM+1
            MUL     B1
            ALSD    1
            ADM     SUM
            LDA     SUM+2
            MUL     B2
            ALSD    1
            ADM     SUM
            LDA     SUM+3
            MUL     B3
            ALSD    1
            ADM     SUM
            LDA     SUM+4
            MUL     B4
            ALSD    1
            ADM     SUM               OUTPUT VALUE FOR TIME N
            BRM     DAOPS             PERFORM D/A OF OUTPUT
            LDX     =077700010,2      SHIFT DATA TO PREPARE
            LDA     DATA,2            FOR SAMPLE TIME N+1
            STA     DATA+1,2
            BRX     $-2,2
*
*
*
*
DATA        RES     5
SUM         RES     5
DAADR       DATA    5                 D/A ADDRESS
A0          DATA    549898
A1          DATA    000000000
A2          DATA    -1099797
A3          DATA    000000000
A4          DATA    549898
B1          DATA    5872294
B2          DATA    -5336061
B3          DATA    2761609
B4          DATA    -1141761
MASK        DATA    077777000
            END     START
```

## COMPUTER PROGRAM 5

```
*   DIGITAL FILTER SIMULATION
*   FOURTH-ORDER BANDPASS FILTER (CASCADE FORM)
*
*
*
*
*
*
       BRM     ADOPS             TAKE SAMPLE
       LDA     SUM
       ARSA    2                 DIVIDE INPUT VALUE BY 4
       STA     SUM+1
       LDX     =077700001,3
       COPY    (0,5)
REPEAT STA     SUM
       LDA     SUM+2,3
       MUL     B1,3
       ALSD    1
       ADM     SUM,3
       LDA     SUM+4,3
       MUL     B2,3
       ALSD    1
       ADM     SUM,3
       LDA     SUM,3
       MUL     A0,3
       ALSD    1
       STA     Y,3
       LDA     SUM+2,3
       MUL     A1,3
       ALSD    1
       ADM     Y,3
       LDA     SUM+4,3
       MUL     A2,3
       ALSD    1
       ADM     Y,3
       LDA     Y,3
       BRX     REPEAT,3
       BRM     DAOPS             PERFORM D/A OF OUTPUT
       LDX     =077600004,2
       LDA     SUM,2
       STA     SUM+2,2
       LDA     SUM+1,2
       BRX     $-4,2
       STA     SUM+3,2
*
*
*
*
*
*
*
*
*
SUM    RES     6
Y      RES     2
DAADR  DATA    5
A0     DATA    1518695
       DATA    1518695
A1     DATA    3037389
       DATA    -3037389
A2     DATA    1518695
       DATA    1518695
B1     DATA    746758
       DATA    5125536
B2     DATA    -1890880
       DATA    -2532626
MASK   DATA    077777000
       END     START
```

COMPUTER PROGRAM 6

```
C       EVALUATION OF RESIDUES IN A PARTIAL FRACTION EXPANSION
C
C       N IS THE ORDER OF THE DENOMINATOR POLYNOMIAL
C       A(I) ARE THE REAL NUMERATOR COEFFICIENTS IN ASCENDING
C       ORDER BEGINING WITH THE CONSTANT TERM
C       P(I) ARE THE COMPLEX POLE FACTORS. THE MULTIPLICITY OF
C       POLES CANNOT EXCEED 2
C       K IS THE COMPLEX RESIDUE ASSOCIATED WITH A GIVEN POLE
C       FACTOR AS COMPUTED BY SUBROUTINE RES
C
        COMPLEX K,P
        DIMENSION A(6),P(6),K(6)
      1 READ(5,100)N
    100 FORMAT(I1)
        IF(N.EQ.0)GO TO 99
        READ(5,101)(A(I),I=1,6)
    101 FORMAT(6F10.0)
        READ(5,102)(P(I),I=1,6)
    102 FORMAT(8F10.0/4F10.0)
        CALL RES(N,A,P,K)
        WRITE(6,200)N,(A(I),I=1,6)
    200 FORMAT(1H1,6X,'THE DEGREE OF THE DENOMINATOR
       *POLYNOMIAL IS ',I1//6X,'THE COEFFICIENTS OF THE
       *NUMERATOR POLYNOMIAL ARE:'//(13X,6F10.6))
        WRITE(6,201)(P(I),K(I),I=1,N)
    201 FORMAT(1H0,16X,'POLE FACTOR',10X,'RESIDUES'//15X,
       *'REAL',6X,'IMAG',6X,'REAL',6X,'IMAG'//(11X,4F10.6))
        GO TO 1
     99 STOP
        END


        SUBROUTINE RES(N,A,P,K)
        COMPLEX K,P,S,Z,NUM,DENOM,FUNCS,FUNCDS,DELTA
        DIMENSION A(6),P(6),K(6)
        NUM(Z)=A1+A2*Z+A3*Z**2+A4*Z**3+A5*Z**4+A6*Z**5
        A1=A(1)
        A2=A(2)
        A3=A(3)
        A4=A(4)
        A5=A(5)
        A6=A(6)
        DELTA=(.001,0.)
        I=1
      5 L=0
      6 S=-P(I)
      7 DENOM=(1.,0.)
        DO 50 J=1,N
        IF(CABS(S+P(J))-.05)9,9,8
      8 DENOM=DENOM*(S+P(J))
        GO TO 50
      9 L=L+1
     50 CONTINUE
        GO TO(11,12,13,13,14,14),L
     11 K(I)=NUM(S)/DENOM
        GO TO 10
     12 K(I)=NUM(S)/DENOM
        I=I+1
        GO TO 6
     13 FUNCS=NUM(S)/DENOM
        S=S+DELTA
        GO TO 7
     14 FUNCDS=NUM(S)/DENOM
        K(I)=(FUNCDS-FUNCS)/DELTA
     10 IF(I.EQ.N) GO TO 90
        I=I+1
        GO TO 5
     90 RETURN
        END
```

COMPUTER PROGRAM 7

```
*   DIGITAL FILTER SIMULATION
*   FOURTH-ORDER BANDPASS FILTER (PARALLEL FORM)
*
*
*
*
*
*
        BRM      ADOPS            TAKE SAMPLE
        LDA      DATA
        ARSA     2                DIVIDE INPUT BY 4
        STA      DATA
        MUL      GAMMA
        ALSD     1
        STA      ANSWER
        LDX      =077700001,3
REPEAT  LDA      DATA
        STA      SUM,3
        LDA      SUM+2,3
        MUL      B1,3
        ALSD     1
        ADM      SUM,3
        LDA      SUM+4,3
        MUL      B2,3
        ALSD     1
        ADM      SUM,3
        LDA      SUM+2,3
        MUL      A1,3
        ALSD     1
        STA      Y,3
        LDA      SUM+4,3
        MUL      A2,3
        ALSD     1
        ADM      Y,3
        LDA      Y,3
        BRX      REPEAT,3
        LDA      Y
        ADM      ANSWER
        LDA      Y+1
        ADM      ANSWER
        LDA      ANSWER
        BRM      ADOPS            PERFORM D/A OF OUTPUT
*
*
*
*
*
*
*
*
*
*
DATA    RES      1
SUM     RES      6
Y       RES      2
ANSWER  RES      1
DAADR   DATA     5                D/A ADDRESS
GAMMA   DATA     549898
A1      DATA     -1609581
        DATA     2379471
A2      DATA     627585
        DATA     194230
B1      DATA     5125536
        DATA     746758
B2      DATA     -2532626
        DATA     -1890880
MASK    DATA     077777000
        END      START
```

## LIST OF REFERENCES

1. Kaiser, J. F., "Digital Filters," in System Analysis by Digital Computer, F. F. Kuo and J. F. Kaiser, Eds. New York: Wiley 1966, ch. 7.

2. Ragazzini, J. R., and Franklin, G. F., Sampled-Data Control Systems, New York: McGraw-Hill, 1958.

3. Kuo, B., Analysis and Synthesis of Sampled-Data Control Systems, Englewood Cliffs, N.J.: Prentice-Hall, 1963.

4. Gold, B., and Rader, C. M., Digital Processing of Signals, New York: McGraw-Hill, 1969.

5. G-AE Concepts Subcommittee, "On Digital Filtering," IEEE Trans. Audio and Electroacoustics, vol. Au-16, pp. 303-314, September 1968.

6. Golden, R. M., "Digital Filter Synthesis by Sampled-Data Transformation," IEEE Trans. Audio and Electroacoustics, vol. AU-16, pp. 321-329, September 1968.

7. Weaver, C. S., and others, "Digital Filtering with Applications to Electrocardiogram Processing," IEEE Trans. Audio and Electroacoustics, vol. AU-16, pp. 350-389, September 1968.

8. Jackson, L. B., Kaiser, J. F., and McDonald, H. S., "An Approach to the Implementation of Digital Filters," IEEE Trans. Audio and Electroacoustics, vol. AU-16, pp. 413-421, September 1968.

9. Nowak, D. J., and Schmid, P. E., "Introduction to Digital Filters," IEEE Trans. Electromagnetic Compatibility, vol. EMC-10, pp. 210-221, June 1968.

10. Kuo, F. F., Network Analysis and Synthesis, New York: Wiley, 1966, ch. 13.

11. Calahan, D. A., "Linear Network Analysis and Realization Digital Computer Programs: An Instruction Manual," University of Illinois Bulletin, vol. 62, no. 58,

12. Special Issue on Fast Fourier Transform and its Application to Digital Filtering and Spectral Analysis, IEEE Trans. Audio and Electroacoustics, vol. AU-15, June 1967.

13. XDS 9300 Computer Reference Manual, Xerox Data Systems Reference Manual, no. 90 00 50F, March 1967.

14. SYMBOL and META-SYMBOL Reference Manual, Xerox Data Systems Reference Manual, no. 90 05 06F, August 1967.

## INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Documentation Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 20 |
| 2. | Library, Code 0212<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 3. | Commandant of the Marine Corps (Code AO3C)<br>Headquarters, U.S. Marine Corps<br>Washington, D.C. 20380 | 1 |
| 4. | James Carson Breckinridge Library<br>Marine Corps Development and Educational Command<br>Quantico, Virginia 22134 | 1 |
| 5. | Professor Rudolf Panholzer<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 6. | Captain Philip J. Walsh, USMC<br>1122 Leahy Road<br>Monterey, California 93940 | 1 |

Unclassified
Security Classification

**DOCUMENT CONTROL DATA - R & D**

*(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)*

| | |
|---|---|
| 1. ORIGINATING ACTIVITY *(Corporate author)*<br>Naval Postgraduate School<br>Monterey, California 93940 | 2a. REPORT SECURITY CLASSIFICATION<br>Unclassified<br>2b. GROUP |

3. REPORT TITLE

A STUDY OF DIGITAL FILTERS

4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)*

Master's Thesis; December 1969

5. AUTHOR(S) *(First name, middle initial, last name)*

Philip Joseph Walsh
Captain, United States Marine Corps

| | | |
|---|---|---|
| 6. REPORT DATE<br>December 1969 | 7a. TOTAL NO. OF PAGES<br>128 | 7b. NO. OF REFS<br>14 |
| 8a. CONTRACT OR GRANT NO.<br>b. PROJECT NO.<br>c.<br>d. | 9a. ORIGINATOR'S REPORT NUMBER(S)<br>9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* | |

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

| | |
|---|---|
| 11. SUPPLEMENTARY NOTES | 12. SPONSORING MILITARY ACTIVITY<br>Naval Postgraduate School<br>Monterey, California 93940 |

13. ABSTRACT

This thesis treats the subject of digital filters in two parts. First the analytical tools and synthesis techniques which are used for digital signal processors are explained and illustrated with examples. Emphasis is placed on 1) the generation and use of the digital transfer function, 2) design procedures in the frequency domain using methods of continuous-filter design as intermediate steps, and 3) real-time digital-filter implementation schemes.

In the second part of the thesis a machine-language computer program is presented for the real-time simulation of digital filters on a hybrid computer. The program is described in detail and experimental results for several filter designs and realization schemes are reported. It is believed that such a computer program, with its inherent ability to accurately simulate a special-purpose computer and to provide external control of word length and clock frequency, could be used to experimentally determine specifications for the LSI implementation of digital filters.

DD FORM 1 NOV 65 1473 (PAGE 1)
S/N 0101-807-6811

Unclassified
Security Classification

A-31408

Unclassified
Security Classification

| 14 KEY WORDS | LINK A | | LINK B | | LINK C | |
|---|---|---|---|---|---|---|
| | ROLE | WT | ROLE | WT | ROLE | WT |
| filter | | | | | | |
| digital filter | | | | | | |
| digital signal processing | | | | | | |
| special-purpose digital computer | | | | | | |

DD FORM 1 NOV 65 1473 (BACK)
S/N 0101-807-6821

Unclassified
Security Classification

A-31409

thesW22283
A study of digital filters.
3 2768 000 99453 7
DUDLEY KNOX LIBRARY